(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2000－322288
（P2000－322288A）
(43)公開日　平成12年11月24日(2000.11.24)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　11/28	３４０	Ｇ０６Ｆ　11/28	３４０Ａ	５Ｂ０４２
9/06	５３０	9/06	５３０Ｔ	５Ｂ０４５
	５４０		５４０Ｕ	５Ｂ０７６
9/44	５１６	9/44	５１６	
15/16	６２０	15/16	６２０Ｔ	

審査請求　未請求　請求項の数10　ＯＬ　（全 34 頁）

(21)出願番号　特願平11－126471

(22)出願日　平成11年５月６日(1999.5.6)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　久保田　康雄
東京都大田区西蒲田七丁目37番10号　株式会社富士通金融システムズ内

(72)発明者　柴　信夫
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

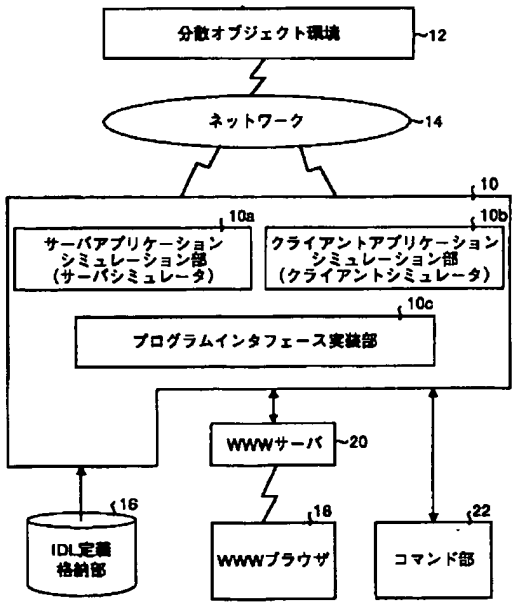
(74)代理人　100089118
弁理士　酒井　宏明

最終頁に続く

(54)【発明の名称】　分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体

(57)【要約】
【課題】　テスト用アプリケーションの開発を不要にし、かつクライアントおよびサーバアプリケーションの開発環境を分離することができる分散オブジェクト開発システムを提供すること。
【解決手段】　シミュレータマシン１０は、ＩＤＬ定義格納部１６からＩＤＬ情報を取得し、取得したＩＤＬ情報に基づいてサーバシミュレータまたはクライアントシミュレータの機能を自動構築する。また、シミュレータマシン１０は、ノードの一つであるWWWサーバ２０と接続され、他のノードは、WWWブラウザ１８を起動してWWWサーバ２０にアクセスすることによって、シミュレータマシン１０が提供するサーバシミュレータまたはクライアントシミュレータの機能を自己のノード上に構築することができる。

本発明にかかる分散オブジェクト開発システムの概略構成図



1

【特許請求の範囲】

【請求項１】　ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムにおいて、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出しまたは前記オブジェクトの実行をシミュレートするアプリケーションシミュレート手段を備えたことを特徴とする分散オブジェクト開発システム。

【請求項２】　ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムにおいて、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成し、生成された応答を前記クライアントアプリケーションに対して送信するサーバアプリケーションシミュレート手段と、

前記インタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出し要求を生成し、生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントアプリケーションシミュレート手段のうち、少なくとも一つを備えたことを特徴とする分散オブジェクト開発システム。

【請求項３】　前記サーバアプリケーションシミュレート手段または前記クライアントアプリケーションシミュレート手段は、前記複数のノードのうちの少なくとも一つのノードに備えられ、当該ノードを除く他のノードのうちの所望のノードが、前記ネットワークを介して、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を制御することを特徴とする請求項２に記載の分散オブジェクト開発システム。

【請求項４】　前記サーバアプリケーションシミュレート手段および前記クライアントアプリケーションシミュレート手段のシミュレート機能を、前記ノードの一つであるWWW（World Wide Web）サーバ上に備え、前記ノードの他の一つがWWWブラウザを介して前記シミュレート機能を制御することを特徴とする請求項３に記載の分散オブジェクト開発システム。

2

【請求項５】　前記サーバアプリケーションシミュレート手段は、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容と、前記生成された応答の内容と、を含む検証用ファイルを作成することを特徴とする請求項２～４のいずれか一つに記載の分散オブジェクト開発システム。

【請求項６】　前記サーバアプリケーションシミュレート手段は、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された応答の内容に付加して、前記検証用ファイルを作成することを特徴とする請求項５に記載の分散オブジェクト開発システム。

【請求項７】　前記クライアントアプリケーションシミュレート手段は、前記生成された要求の内容と、該要求に応じて前記サーバアプリケーションから送信された応答の内容と、を含む検証用ファイルを作成することを特徴とする請求項２～４のいずれか一つに記載の分散オブジェクト開発システム。

【請求項８】　前記クライアントアプリケーションシミュレート手段は、前記サーバアプリケーションから送信された応答の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された要求の内容に付加して、前記検証用ファイルを作成することを特徴とする請求項７に記載の分散オブジェクト開発システム。

【請求項９】　ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、

取得したインタフェース情報に基づいて、前記オブジェクトの呼び出しおよび／または前記オブジェクトの実行をシミュレートするシミュレート手順と、

を実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項１０】　ネットワーク上の複数のノードにオブ

ジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記サーバアプリケーションと前記クライアントアプリケーションのいずれかのシミュレートをおこなうかを選択する選択手順と、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、

前記選択ステップにおいて前記サーバアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成する応答生成手順と、

前記応答生成手順において生成された応答を前記クライアントアプリケーションに対して送信するサーバシミュレート手順と、

前記選択手順において前記クライアントアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記オブジェクトの呼び出し要求を生成する要求生成手順と、

前記要求生成手順において生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントシミュレート手順と、

を実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ネットワーク上に散りばめられたオブジェクトが相互に連携しながら全体として機能する分散オブジェクトシステムにおいて、特に、オブジェクト指向プログラミングによるクライアント／サーバアプリケーションの開発を支援するための分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体に関する。

【０００２】

【従来の技術】近年、通信インフラの整備の向上と、ＬＡＮ（Local Area Network）やインターネットの普及によって、ネットワークを介して所望のサービスを享受することのできる分散コンピューティングの発展が目覚ましい。従来からの集中処理型のホストコンピュータ方式においては、データベースの利用や複雑な計算を必要とする処理を高性能なホストコンピュータがおこなうことによって、これに接続された複数の端末が所望のサービスを享受できるという形態であった。

【０００３】このホストコンピュータ方式に代わる分散コンピューティングは、近年の低価格かつ高性能なパーソナルコンピュータ（ＰＣ）やワークステーション（ＷＳ）を用いて、サービスの提供を高信頼性および大量かつ高速処理に向いたサーバ・マシンでおこない、データの加工やユーザインタフェースを上記したＰＣやＷＳなどのクライアント・マシンでおこない、さらにこれらをネットワークで接続することにより、処理の分散化を図った、いわゆるクライアント／サーバ・システムを実現したものである。

【０００４】分散コンピューティングを実現するネットワークプログラミングは、通常、ＲＰＣ（リモートプロシジャコール）に基づいておこなわれている。ＲＰＣは、ネットワーク上の異なる場所にあるプロシジャ（関数など）の呼び出しをおこなうもので、ネットワークを利用するプロシジャを各サーバにライブラリとして用意しておき、これを通してネットワークの利用を可能とするプログラミングの形態である。

【０００５】よって、そのプロシジャのライブラリを整備しておけば、ネットワーク構成、プロトコルおよびプロセス間通信を意識しないでネットワーク・プログラミングをおこなうことが可能となる。このＲＰＣによって、サーバ上のプロシジャがあたかもクライアントにあるかのように見せかけて、ネットワーク間のデータ転送とマシン間のデータ変換を隠蔽することができる。

【０００６】ところが、このような分散コンピューティングにおいて、ネットワーク規模の拡大や上記したサービスを提供するためのアプリケーションの高機能化に伴って、通信トラフィックやサーバの負荷の増大、またはアプリケーション開発の複雑化が問題となっている。特にこれまでは、アプリケーションの開発を、オペレーティングシステム（ＯＳ）毎に異なる開発言語や開発ツールなどを使用することを中心に発展させてきたが、ネットワーク化が普及してくると、これまでのアプリケーション開発と、通信、リソース管理、障害対策などの実装環境の構築とを別々におこなってゆくやり方では、アプリケーション開発者に大きな負担と多くの困難を与えることになる。

【０００７】そこで、これらの問題を解決するべく、処理単位のさらなる分散化とネットワークプログラミングにおけるアプリケーションコンポーネントの再利用化を図った分散オブジェクト技術が注目されている。ここで、オブジェクトとは、データと処理手続き（メソッドまたはオペレーションと呼ぶ）が一体となったプログラムを意味する。分散オブジェクトシステムでは、ある処

理に対して、オブジェクト同士が動的にクライアント／サーバの関係を構築して処理をおこなうことができ、ネットワーク全体が一つの巨大なコンピュータとして機能することができる。

【０００８】上記した分散オブジェクトのアーキテクチャとして代表的なものの一つに、米国のオブジェクト指向技術の標準化団体であるＯＭＧ（The Object ManagementGroup ）が策定したＣＯＲＢＡ（共通オブジェクト・リクエスト・ブローカ・アーキテクチャ）がある。以下に、このＣＯＲＢＡが提供する分散オブジェクト管理の標準仕様について説明する。

【０００９】元来、ネットワーク上に分散したアプリケーションコンポーネントを共有して使えるようするためのフレームワークとして、ＯＲＢ（オブジェクト・リクエスト・ブローカ）という相互コミュニケーションの方式が提案されていた。ＯＲＢでは、ネットワーク上のサービスを実行するプログラムをオブジェクトとみなし、クライアントプログラム（すなわちクライアントオブジェクト）は、このＯＲＢを介して、同一マシン上またはネットワーク上にあるサーバオブジェクトのメソッドを、それがどこにあるかを意識することなく呼び出すことができる。

【００１０】また、ＯＲＢは、オブジェクトがネットワーク上のどのマシンに存在するのかを管理しており、サービスの呼び出し時にマシン名を指定する必要はないため、ＯＲＢを利用したアプリケーションにおいて、あるサービスを別のマシンに移動するような場合でもプログラムの変更を必要とせずにＯＲＢ上での設定を変えるだけですむようになり、アプリケーションの柔軟性を向上させることが可能となる。

【００１１】よって、クライアントは、ＯＲＢを仲介して、利用したいオブジェクトに対してリクエストを送り、リクエストを受け取ったオブジェクトは処理を実行し、再びＯＲＢを介して結果をクライアントに返すことになる。ここで、クライアントとは、オブジェクトを利用する側のことであり、サーバは、その利用されるオブジェクト（一般のオブジェクトと区別するため、以下において、オブジェクト・インプリメンテーションと呼ぶ）を提供する側のことである。オブジェクト・インプリメンテーションが別のサーバ・オブジェクトに対するクライアントになることも可能であるため、ＯＲＢ上のオブジェクトは、その時々でクライアントになったりサーバになったりする。

【００１２】また、ＯＲＢ上の分散オブジェクトはマシンやシステムだけでなく、プログラミング言語にも依存しない形で記述できなければならないため、ＩＤＬ（インタフェース定義言語）と呼ばれる言語を導入することでこの問題の解決を図っている。ＩＤＬは他の言語と違ってインタフェースの定義のみをおこなう専用の言語であり、オペレーション、例外、および属性のセットを宣言し、各オペレーションは、名前、パラメータ、結果、および例外を定義したシグニチャによって構成される。

【００１３】ＣＯＲＢＡは、以上に説明したＯＲＢの標準化仕様であり、特に、クライアントからＯＲＢにリクエストを送る際に、クライアント側プログラム（クライアントアプリケーション）の言語（ＣやＣ＋＋）と、ＯＲＢを結びつけてメッセージのやり取りを可能にする、いわゆるバインディングの方法として静的起動と動的起動の２種類が用意されていることを特徴の一つとしている。ＣＯＲＢＡのオブジェクトを設計するには、最初にＩＤＬを用いてオブジェクトの公開される機能のみを定義し、それを元にして、それぞれのプログラミング言語の実装がおこなわれる。

【００１４】図２６は、ＣＯＲＢＡにおけるＯＲＢのインタフェース構造を示すブロック図である。図２６に示すＣＯＲＢＡのインタフェース構造は、ＯＲＢコア２６００、動的起動インタフェース２６５０、クライアントスタブ２６６０、ＯＲＢインタフェース２６７０、オブジェクトアダプタ２６８０、サーバスケルトン２６９０、インタフェースリポジトリ２６３０およびインプリメンテーションリポジトリ２６４０から構成されている。

【００１５】図２６において、ＯＲＢコア２６００は、クライアント２６１０とサーバ（オブジェクト・インプリメンテーション）２６２０との間を位置に依存しない通信をつかさどるもので、一般にオブジェクトバスを定義するものとして解釈されている。このオブジェクトバスは、インタフェースリポジトリ２６３０、インプリメンテーションリポジトリ２６４０またはその他ＣＯＲＢＡサービスによって拡張することができる。

【００１６】動的起動インタフェース２６５０は、上記した動的起動をおこなう際、実行時に、連携する先のオブジェクト（以下、ターゲットオブジェクトと呼ぶ）の型情報を取り出して、ターゲットオブジェクトの呼び出しをおこなうものであり、この動的起動のインタフェース仕様は各オブジェクト共通である。

【００１７】また、クライアントスタブ２６６０は、ターゲットオブジェクトのインタフェース定義がマッピングされた関数であり、クライアント２６１０がサーバ２６２０上においてターゲットオブジェクトを起動するための方法を定義するものである。よって、クライアントスタブ２６６０は、上記した静的起動をおこなうために、あらかじめクライアントアプリケーションの作成時にターゲットオブジェクトの型ごとに生成されて、静的起動をおこなう際、ターゲットオブジェクトの呼び出しをおこなうものである。

【００１８】ＯＲＢインタフェース２６７０は、ＯＲＢに対する操作を行うためのインタフェースであり、アプリケーションが必要とするローカルなサービスに対する少数のＡＰＩ（Application Programming Interface ）

から構成され、たとえば、オブジェクトを参照するためのオブジェクト・リファレンスを文字列に変換したり、その逆の変換をおこなうためのＡＰＩを提供している。

【００１９】オブジェクトアダプタ２６８０は、オブジェクト・インプリメンテーションに代わって、サービスの要求を受け付け、オブジェクト・インプリメンテーションの生成、オブジェクト・インプリメンテーションへの要求伝達、オブジェクト・リファレンスを割当てるための実行時環境を提供するものである。特に、ＯＲＢでは、ターゲットオブジェクトへの最初の呼び出しがおこなわれると、プロセスが生成される（オブジェクトの活性化）が、オブジェクトアダプタ２６８０は主にこのオブジェクトの活性化および非活性化をおこなう。

【００２０】サーバスケルトン２６９０は、サーバ２６２０によってエクスポートされるそれぞれのサービスへの静的なインタフェースの提供、すなわちクライアントスタブ２６６０が変換した構造を元の関数呼び出し形式に復元して、ターゲットオブジェクトの実装コードの呼び出しをおこなうもので、クライアントスタブ２６６０とともにＩＤＬコンパイラを使って生成される。

【００２１】なお、ここでは、図示していないが、クライアント側の動的起動インタフェース２６５０と同等の機能をサーバ側に提供する動的スケルトンインタフェースも、上記したサーバスケルトン２６９０に含むものとする。この動的スケルトンインタフェースは、クライアントアプリケーションの作成時に、ターゲットオブジェクトの実装型が不明な場合に、リクエストをＯＲＢからオブジェクト・インプリメンテーションに渡す方法を提供するＡＰＩである。

【００２２】インタフェースリポジトリ２６３０は、動的起動の際に、実行時に取り出されるＩＤＬで作成されたインタフェース情報をオブジェクトの形で格納したものであり、インプリメンテーションリポジトリ２６４０は、サーバ２６２０がサポートするクラス、生成されたオブジェクト、およびそれらの実行時リポジトリを提供するものである。

【００２３】つぎに、以上に説明したＣＯＲＢＡ仕様に準拠した分散オブジェクトシステムにおけるクライアントアプリケーションの開発について説明する。図２７は、従来のクライアントアプリケーション開発手順を示すフローチャートである。図２７に示すフローチャートは、特に、上記した静的起動をおこなうクライアントアプリケーションの開発をおこなう手順について示している。

【００２４】図２７において、まず、クライアントアプリケーション開発者は、開発しようとするアプリケーションのプログラム・インタフェース設計をおこなう（ステップＳ２７０１）。つづいてクライアントアプリケーションにおいて呼び出される関数のためのＩＤＬファイル、すなわちオブジェクトクラスを作成する（ステップＳ２７０２）。ＩＤＬは、オブジェクトがクライアントの候補に対して、どのようなオペレーションが利用でき、どのようにそれを起動すべきかを教える手段であり、オブジェクトの型、それらの属性、それらがエクスポートするメソッド、およびメソッドのパラメータを定義する。

【００２５】そして、上記したオブジェクトクラスを用いて実行される所望のクライアントアプリケーションのソースを作成する（ステップＳ２７０３）。ここで、このクライアントアプリケーションが実行される際には、上記したオブジェクトクラスを提供するサーバが必要となるため、ステップＳ２７０３において作成したクライアントアプリケーションの実行を検証する場合、すなわちテストする場合には、サーバアプリケーションが必要となる。

【００２６】しかしながら、プログラムの実行検証をおこなう段階において、実際にネットワーク上で稼動しているサーバアプリケーションを用いることは、テストされるクライアントアプリケーションによってそのサーバアプリケーションが破壊されたり、ネットワーク上の他のオブジェクトに不具合を与えたりすることが考えられるため、現実的ではない。

【００２７】そこで、通常は、テスト用のサーバアプリケーションを作成し（ステップＳ２７０４）、このテスト用サーバアプリケーションを起動させた環境において、作成したクライアントアプリケーションの実行検証をおこなう。

【００２８】つぎに、ステップＳ２７０２において作成したＩＤＬファイルを、ＩＤＬコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびＩＤＬオブジェクトを生成する（ステップＳ２７０５）。ここで、ＩＤＬオブジェクトは、インタフェースリポジトリに結合され（ステップＳ２７０６）、動的起動をおこなう際のＩＤＬ情報となる。

【００２９】そして、ステップＳ２７０３において作成されたクライアントアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップＳ２７０５において生成されたクライアントスタブをリンクすることで、実行可能なクライアントアプリケーションのファイルを得る（ステップＳ２７０７）。

【００３０】また、同様に、ステップＳ２７０４において作成されたテスト用サーバアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップＳ２７０５において生成されたサーバスケルトンをリンクすることにより、実行可能なテスト用サーバアプリケーションのファイルを得る（ステップＳ２７０８）。

【００３１】ステップＳ２７０７によって得たテスト用サーバアプリケーションは、インプリメンテーションリポジトリに記録されるとともに、クライアントアプリケ

ーションによって呼び出し可能なオブジェクト・インプリメンテーションとして登録される（ステップＳ２７０９）。

【００３２】そして、作成したクライアントアプリケーションの実行検証をおこなうために、クライアントから送信されたリクエストに対して応答する電文（テキストイメージ）となるテストデータを作成し（ステップＳ２７１０）、ステップＳ２７０９において登録されたテスト用サーバアプリケーションの起動をおこなう（ステップＳ２７１１）。この状態で、ステップＳ２７０７において生成されたクライアントアプリケーションを実行し、その結果を検証する（ステップＳ２７１２）。

【００３３】一方、サーバアプリケーションの開発をおこなう場合も上述したクライアントアプリケーションの開発と同様に、テスト用クライアントアプリケーションを作成する必要がある。図２８は、従来のサーバアプリケーション開発手順を示すフローチャートである。図２８に示すフローチャートにおいても、静的起動をおこなうサーバアプリケーションの開発をおこなう手順について示している。

【００３４】図２８において、まず、サーバアプリケーション開発者は、開発しようとするサーバアプリケーションのプログラム・インタフェース設計をおこなう（ステップＳ２８０１）。つづいてサーバアプリケーションにおいて呼び出される関数のためのＩＤＬファイル、すなわちオブジェクトクラスを作成する（ステップＳ２８０２）。

【００３５】そして、上記したオブジェクトクラスを用いて実行される所望のサーバアプリケーションのソースと、テスト用のクライアントアプリケーションのソースとを作成する（ステップＳ２８０３、ステップＳ２８０４）。つづいて、ステップＳ２８０２において作成したＩＤＬファイルを、ＩＤＬコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびＩＤＬオブジェクトを生成し（ステップＳ２８０５）、ＩＤＬオブジェクトを、インタフェースリポジトリに結合する（ステップＳ２８０６）。

【００３６】そして、ステップＳ２８０３において作成されたサーバアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップＳ２８０５において生成されたサーバスケルトンをリンクすることにより、実行可能なサーバアプリケーションのファイルを得る（ステップＳ２８０７）。

【００３７】また、同様に、ステップＳ２８０４において作成されたテスト用クライアントアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップＳ２８０５において生成されたクライアントスタブをリンクすることで、実行可能なテスト用クライアントアプリケーションのファイルを得る（ステップＳ２８０８）。

【００３８】ステップＳ２８０８において得られたサーバアプリケーションは、インプリメンテーションリポジトリに記録されるとともに、テスト用クライアントアプリケーションによって呼び出し可能なオブジェクト・インプリメンテーションとして登録される（ステップＳ２８０９）。

【００３９】そして、サーバアプリケーションの実行検証をおこなうために、テスト用クライアントアプリケーションからサーバアプリケーションに送信されるリクエストとなるテストデータ（テキストイメージ）を作成し（ステップＳ２８１０）、ステップＳ２８０９において登録されたサーバアプリケーションを起動する（ステップＳ２８１１）。この状態で、ステップＳ２８０７において生成されたテスト用クライアントアプリケーションを実行し、その実行に応じて得られるサーバアプリケーションの動作結果を検証する（ステップＳ２８１２）。

【００４０】以上に説明したように、分散オブジェクトシステムにおいては、クライアント／サーバ・モデルを基本構造としているため、通常、アプリケーションの実行において、対となるクライアントまたはサーバを必要とし、アプリケーションの開発をおこなう段階にあっても、作成したアプリケーションの実行検証をするために、対となるクライアントまたはサーバ上で動作するアプリケーションが必要となる。

【００４１】また、他の分散オブジェクト・システムにおけるアプリケーションの開発方法として、たとえば特開平９－１２０３６６号公報に開示の「分散アプリケーション・プログラムをデバッグする分散デバッグのためのシステムおよび方法」が提案されている。この「分散アプリケーション・プログラムをデバッグする分散デバッグのためのシステムおよび方法」によれば、ＣＯＲＢＡ準拠の分散オブジェクト環境において、ローカル・コンピュータおよび遠隔コンピュータにデバッガ・エンジンを常駐させ、これらローカル・コンピュータおよび遠隔コンピュータのうちの少なくとも一つにデバッガＧＵＩを備えている。

【００４２】そして、このデバッガＧＵＩが通信機構によってデバッガ・エンジンと通信することにより、クライアントアプリケーションがローカル・ホストでデバッガを使用できるとともに、オブジェクトを含んでいるアプリケーションおよび未知の遠隔ホスト・コンピュータで作動するオブジェクト・インプリメンテーションをシームレスにデバッグすることが可能となっている。

【００４３】

【発明が解決しようとする課題】しかしながら、ＣＯＲＢＡに準拠した分散オブジェクトシステムでは、通信相手となるアプリケーション（オブジェクト）の場所およびインタフェースを認識しているのは、開発者ではなくＯＲＢであり、クライアントアプリケーション開発者がサーバアプリケーション開発者でもあるという特別な場

合を除いては、未知のホストシステム上に配置される未知のオブジェクトとの通信が要求される。

【００４４】すなわち、ＣＯＲＢＡ準拠システムにおいては、クライアントアプリケーションがターゲットオブジェクトの配置位置を直接取得することなく、ターゲットオブジェクトの使用を可能にしていることを特徴としており、一般に、ＣＯＲＢＡ準拠システムのアプリケーション開発者は自分のオブジェクトに関連したサーバを探し出すことができない。

【００４５】よって、ＣＯＲＢＡ仕様に準拠するような分散オブジェクトシステム上で動作させるアプリケーションを開発するには、上述したように、目的とするクライアントアプリケーションまたはサーバアプリケーションのソース作成とは別に、テスト用のクライアントまたはサーバアプリケーションを作成する必要があった。このことは、目的とするアプリケーションの実行検証の信頼性を高めるためにも、テスト用のアプリケーションにバグが含まれることは許されないため、アプリケーション開発者に、多大な負担を与える結果となっていた。

【００４６】また、ＣＯＲＢＡ仕様に準拠する分散オブジェクトシステムは、アプリケーションの言語依存性、ＯＳ依存性を排除するために上述したＩＤＬを導入している一方で、アプリケーション開発者にアプリケーションのプログラム言語とは別にＩＤＬという言語の習得を要請するものであり、作成したアプリケーションの実行検証においてもこのＩＤＬの入出力が正しくおこなわれているかどうかを検証する必要があるが、図２７および図２８に示したような従来のアプリケーション開発手順では、クライアントおよびサーバの開発環境が分離されていないため、ネットワークを介した実際の分散オブジェクト環境上、特にＯＲＢ上におけるアプリケーションの動作を完全に再現することはできず、実行検証をおこなうシステム（ホストシステム）に依存したアーキテクチャ（ＯＳ等）に従って、クライアントのリクエストまたはサーバの応答を作成しなければならなかった。

【００４７】また、上記した特開平９－１２０３６６号公報に開示の「分散アプリケーション・プログラムをデバッグする分散デバッグのためのシステムおよび方法」では、デバッグ環境を実際の分散オブジェク環境上で実現しているために、より信憑性の高いアプリケーションの実行検証をおこなうことができる一方で、バグの存在によって、ネットワークに接続された他のノード（他のクライアントやサーバ）に影響を与える可能性がある。

【００４８】さらに、実行検証させるサーバまたはクライアントアプリケーションと対になるサーバまたはクライアントアプリケーションのマシンにおいてもデバッガ・マシンを搭載する必要があるが、ＣＯＲＢＡ仕様のように、ターゲットオブジェクトの位置を正確に知得できない場合、特に動的起動をおこなうような場合に、ターゲットオブジェクトを備えたサーバのそれぞれにデバッガ・マシンを搭載することは現実的ではない。

【００４９】本発明は、上記に鑑みてなされたものであって、サーバアプリケーションに対してクライアントアプリケーション、クライアントアプリケーションに対してサーバアプリケーションのように、実行検証の対象となるアプリケーションに対して対となるアプリケーションをシミュレートする各シミュレータを備え、テスト用アプリケーションの開発を不要にし、クライアントおよびサーバアプリケーションの開発環境を分離するとともに、所望のホストシステム上にアプリケーション開発環境を構築することができる分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体を提供することを目的とする。

【００５０】

【課題を解決するための手段】上述した課題を解決し、目的を達成するために、請求項１の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムは、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出しまたは前記オブジェクトの実行をシミュレートするアプリケーションシミュレート手段（後述するサーバシミュレータまたはクライアントシミュレータに相当する）を備えているので、アプリケーションの実行検証をおこなう際に対となるテスト用のアプリケーションの実行に必要なテストデータ（後述する情報ファイルに相当する）の自動生成が可能となり、このテストデータをオブジェクトの呼び出しおよび／またはオブジェクトの実行をおこなう際の通信電文として用いることで、サーバシミュレータまたはクライアントシミュレータを自動的に構築することができる。

【００５１】また、請求項２の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムは、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、

取得した情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答（後述する復帰情報に相当する）を生成し、生成された応答を前記クライアントアプリケーションに対して送信するサーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）を備えているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【００５２】また、前記インタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出し要求（後述する起動情報に相当する）を生成し、生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）を備えているので、クライアントアプリケーションをシミュレートする機能を自動的に構築することができる。

【００５３】また、請求項３の発明において、請求項２に記載の分散オブジェクト開発システムは、前記複数のノードのうちの少なくとも一つのノードが前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）または前記クライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）を備えており、当該ノードを除く他のノードのうちの所望のノードが、前記ネットワークを介して、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を制御するので、テストの対象となるアプリケーションと、シミュレートされるアプリケーションとをネットワーク上に分離して配置することができる。

【００５４】また、請求項４の発明において、請求項３に記載の分散オブジェクト開発システムは、前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）および前記クライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）のシミュレート機能（後述するシミュレートプログラム）を、前記ノードの一つであるWWW（World Wide Web）サーバ上に備え、前記ノードの他の一つがWWWブラウザを介して前記シミュレート機能を制御するので、WWWブラウザ上のＧＵＩ表示による操作を提供することができる。

【００５５】また、請求項５の発明において、請求項２～４のいずれか一つに記載の分散オブジェクト開発システムは、前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容（後述する起動情報に相当する）と、前記生成された応答の内容（後述する実行履歴に相当する）と、を含む検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したクライアントアプリケーションの設計の誤りを検証することができる。

【００５６】また、請求項６の発明において、請求項５に記載の分散オブジェクト開発システムは、前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容（後述する起動情報に相当する）が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された応答の内容（後述する実行履歴に相当する）に付加して、前記検証用ファイルを作成するので、検証用ファイルを参照することで、開発したクライアントアプリケーションの設計の誤りを容易に発見することができる。

【００５７】また、請求項７の発明において、請求項２～４のいずれか一つに記載の分散オブジェクト開発システムは、前記クライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）が、前記生成された要求の内容（後述する実行履歴に相当する）と、該要求に応じて前記サーバアプリケーションから送信された応答の内容（後述する復帰情報に相当する）と、を含む検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したサーバアプリケーションの設計の誤りを検証することができる。

【００５８】また、請求項８の発明において、請求項７に記載の分散オブジェクト開発システムは、前記クライアントアプリケーションシミュレート手段が、前記サーバアプリケーションから返信された応答の内容（後述する復帰情報に相当する）が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された要求の内容（後述する実行履歴に相当する）に付加して、前記検証用ファイルを作成するので、開発したサーバアプリケーションの設計の誤りを容易に発見することができる。

【００５９】また、請求項９の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体は、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、取得したインタフェース情報に基づいて、前記オ

ブジェクトの呼び出しおよび／または前記オブジェクトの実行をシミュレートするシミュレート手順と、を実行させるためのプログラムを記録しているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【００６０】また、請求項１０の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体は、前記サーバアプリケーションと前記クライアントアプリケーションのいずれかのシミュレートをおこなうかを選択する選択手順と、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、を実行させるためのプログラムを記録しているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【００６１】また、前記選択手順において前記サーバアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成する応答生成手順と、前記応答生成手順において生成された応答を前記クライアントアプリケーションに対して送信するサーバシミュレート手順と、前記選択手順において前記クライアントアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記オブジェクトの呼び出し要求を生成する要求生成手順と、前記要求生成手順において生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントシミュレート手順と、を実行させるためのプログラムを記録しているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【００６２】

【発明の実施の形態】以下に、分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体の実施の形態を図面に基づいて詳細に説明する。なお、この実施の形態によりこの発明が限定されるものではない。

【００６３】図１は、本発明にかかる分散オブジェクト開発システムの概略構成を示すブロック図である。図１において、分散オブジェクト開発システムは、ＯＭＧ仕様によるＯＲＢ間通信用のネットワークプロトコルであるＩＩＯＰ等の通信プロトコルによって構築されたネットワーク１４に、それぞれオブジェクトを分散して備えた複数のノード（ＰＣ、ＷＳ、ホストコンピュータ等）が接続されることで構築された、ＣＯＲＢＡ等に準拠した分散オブジェクト環境１２の下にあって、上記したノードの一つとしてネットワーク１４に接続されたシミュレータマシン１０と、シミュレータマシン１０においてクライアントシミュレータおよびサーバシミュレータの機能を実現するために用いられるＩＤＬ定義格納部１６と、を備えて構成されている。

【００６４】シミュレータマシン１０は、さらに、ＩＤＬ定義格納部１６から取得されるＩＤＬ情報に基づいて自動生成されるテスト用サーバアプリケーション、すなわちサーバシミュレータの機能を提供するサーバアプリケーションシミュレーション部１０ａと、ＩＤＬ定義格納部１６から取得されるＩＤＬ情報に基づいて自動生成されるテスト用クライアントアプリケーション、すなわちクライアントシミュレータの機能を提供するクライアントアプリケーションシミュレーション部１０ｂと、上記したテスト用サーバアプリケーションおよびテスト用クライアントアプリケーションの自動構築およびシミュレータマシン１０に接続される外部装置（上記したノードを含む）の通信制御を担うプログラムインタフェース実装部１０ｃと、を備えている。

【００６５】ここで、ＩＤＬ定義とは、上述したＩＤＬ情報を意味し、クライアントアプリケーションおよびサーバアプリケーションのインタフェースを定義するものであり、ＩＤＬ定義格納部１６は、上記したノードの一つとしてネットワーク１４上に配置されるか、またはシミュレータマシン１０の外部デバイスとして接続されたものである。

【００６６】また、シミュレータマシン１０は、ユーザによってコマンド部２２を介して操作され、このコマンド部２２によって与えられた種々の命令に従って動作することが可能である。さらに、シミュレータマシン１０は、ノードの一つであるＷＷＷ（World Wide Web）サーバ２０と接続されており、他のノード上で起動されるＷＷＷブラウザ１８がこのＷＷＷサーバ２０にアクセスすることにより、ＷＷＷブラウザ１８上に、シミュレータマシン１０が提供する機能（サーバシミュレータ、クライアントシミュレータ）を享受することができる。

【００６７】すなわち、シミュレータマシン１０は、コマンド部２２以外にも、遠隔にあるノードからネットワーク１４を介して操作することができる。特に、ＷＷＷブラウザ１８を利用する場合、上記した機能は、ＷＷＷサーバ２０によってＷＷＷブラウザ１８上にサービスを提供する外部プログラムとして実現される。この場合、ＷＷＷサーバ２０は、ＣＧＩ（Common Gateway Interfa

ce）やＡＰＩを経由することによって外部プログラム、すなわちシミュレータマシン１０の機能を利用することができる。

【００６８】よって、本発明にかかる分散オブジェクト開発システムにおいて、シミュレータマシン１０は、アプリケーション開発者がクライアントアプリケーションの開発をおこなう際には、そのクライアントアプリケーションの実行検証をおこなうために必要なテスト用サーバアプリケーションをシミュレートし、サーバアプリケーションの開発をおこなう際には、そのサーバアプリケーションの実行検証をおこなうために必要なテスト用クライアントアプリケーションをシミュレートすることができる。

【００６９】さらに、このシミュレート機能は、WWWサーバ２０を介して、所望のノードから実行することができる。このことは、インターネットのような広範囲に亘るネットワークに接続可能なノード上において、アプリケーションの開発をおこなうことができ、アプリケーション開発者が使用するノード毎に上記したシミュレート機能を有するプログラムをインストールする必要がなくなることを意味する。

【００７０】また、分散オブジェクト環境において動作するアプリケーションのように、本来、分散されたノードのいずれかに配置されたターゲットオブジェクトを呼び出すアプリケーション、すなわちクライアントアプリケーションや、ターゲットオブジェクトを備えてクライアントアプリケーションから呼び出されるアプリケーション、すなわちサーバアプリケーションについての実行検証をおこなうには、ネットワークを介して遠隔にあるノードとの通信が必要となるが、本発明にかかる分散オブジェクト開発システムでは、ＩＤＬ定義格納部１６からターゲットオブジェクトについてのＩＤＬ情報を取得することにより、そのターゲットオブジェクトによるリクエストの送受信をシミュレートしたテスト用アプリケーションを自己のノードに自動的に構築することができる。

【００７１】以下において、この分散オブジェクト開発システムを用いて、クライアントアプリケーションおよびサーバアプリケーションの開発手順について説明し、各アプリケーションの開発をおこなうにあたり、上記したサーバアプリケーションシミュレーション部（以下、サーバシミュレータと呼ぶ）およびクライアントアプリケーションシミュレーション部（以下、クライアントシミュレータと呼ぶ）の処理についてそれぞれ、WWWブラウザ１８上で実現した場合に表示される画面とともに説明する。

【００７２】（クライアントアプリケーションの開発手順）まず、図２７に示した従来のクライアントアプリケーション開発手順に対応させて、本発明にかかる分散オブジェクト開発システムにおけるサーバシミュレータを用いたクライアントアプリケーションの開発手順について説明する。図２は、本発明にかかる分散オブジェクト開発システムによるクライアントアプリケーション開発手順を示すフローチャートである。図２に示すフローチャートは、特に、上記した静的起動をおこなうクライアントアプリケーションの開発をおこなう手順について示している。

【００７３】図２において、まず、クライアントアプリケーション開発者は、開発しようとするアプリケーションのプログラム・インタフェース設計をおこなう（ステップＳ２０１）。つづいてクライアントアプリケーションにおいて呼び出される関数のためのＩＤＬファイルを作成する（ステップＳ２０２）。

【００７４】そして、上記したＩＤＬファイルを用いて実行される所望のクライアントアプリケーションのソースを作成する（ステップＳ２０３）。つぎに、ステップＳ２０２において作成したＩＤＬファイルを、ＩＤＬコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびＩＤＬオブジェクトを生成する（ステップＳ２０４）。生成されたＩＤＬオブジェクトは、インタフェースリポジトリに結合される（ステップＳ２０５）。

【００７５】つづいて、ステップＳ２０３において作成されたクライアントアプリケーションのソースをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップＳ２０４において生成されたクライアントスタブをリンクすることで、実行可能なクライアントアプリケーションのファイルを得る（ステップＳ２０６）。

【００７６】そして、上記したサーバシミュレータを起動する前処理を実行し（ステップＳ２０７）、その後、サーバシミュレータを起動する処理を実行する（ステップＳ２０８）。これらサーバシミュレータ起動前処理およびサーバシミュレータ起動処理が、本発明にかかる分散オブジェクト開発システムによって実現される処理に相当し、これら処理の詳細については後述する。

【００７７】サーバシミュレータを起動させた後は、ステップＳ２０６において実行可能なファイルとして生成されたクライアントアプリケーションを実行させる（ステップＳ２０９）。ここで実行されるクライアントアプリケーションは、分散オブジェクト環境に実際に搭載する形態と同一のコードを有するものであり、特にテスト用のために変更を加える必要はない。すなわち、ＣＯＲＢＡ上で問題なく動作可能なバイナリソースのクライアントアプリケーションを実行させる。そして、クライアントアプリケーションの実行結果の検証をおこなう（ステップＳ２１０）。

【００７８】図３は、このクライアントアプリケーションの実行処理を説明するためのフローチャートであり、特に静的起動をおこなうためのクライアントスタブがおこなう処理を示している。図３において、まずクライア

ントアプリケーションは、初期化がおこなわれ、必要に応じてオブジェクトの呼び出しリクエストをクライアントスタブに送出する（ステップＳ３０１）。ＣＯＲＢＡ仕様では、このオブジェクト、すなわちターゲットオブジェクトの呼び出しをすべてＯＲＢ上のオブジェクト・リファレンスと呼ばれる参照ＩＤによっておこなっているため、このターゲットオブジェクトのオブジェクト・リファレンスを取得する必要がある。

【００７９】しかし、クライアントアプリケーションは、ターゲットオブジェクトを文字列からなるオブジェクト名を用いて指定しているため、このオブジェクト名に対応したオブジェクト・リファレンスを取得しなければならない。ＣＯＲＢＡには、オブジェクト名からそれに対応したオブジェクト・リファレンスを提供するネーミングサービスが備わっているので、通常は、このネーミングサービスを利用する。

【００８０】ただし、ネーミングサービスもまたＣＯＲＢＡ上のオブジェクトの一つであるため、あらかじめこのネーミングサービスのオブジェクト・リファレンスを取得する必要がある（ステップＳ３０２）。これにより、ネーミングサービスへのアクセスが可能となり、ターゲットオブジェクト（広義ではサーバアプリケーション）のオブジェクト・リファレンスの取得が可能となる（ステップＳ３０３）。

【００８１】そして、クライアントスタブは、ステップＳ３０３において取得したターゲットオブジェクトのオブジェクト・リファレンスを用いて、ＯＲＢコアに送出するリクエストを生成する（ステップＳ３０４）。オブジェクトアダプタは、このリクエストを受け取り、リクエストに含まれるターゲットオブジェクトに対応するサーバスケルトンによって、オブジェクトの呼び出しがおこなわれる（ステップＳ３０５）。

【００８２】実際には、ステップＳ３０５の後、サーバスケルトンが、呼び出したオブジェクトから返信データを受け取り、ＯＲＢコアを介して再び、リクエスト発信元のクライアントに対してその返信データを送出するが、本発明にかかる分散オブジェクト開発システムでは、この処理を、上記したサーバシミュレータがおこなう。よって、サーバシミュレータは、このサーバスケルトンの機能をシミュレートすることにもなる。

【００８３】図３に示したクライアントアプリケーションが実行された後は、サーバシミュレータにより、クライアントに対して返信データが送出され、クライアントアプリケーションの動作結果の検証がおこなわれる（図２のステップＳ２１０）。具体的には、クライアントアプリケーションが送信したリクエストと、サーバシミュレータが受信したデータとを比較することにより、クライアントアプリケーションの設計の誤りを検出する。

【００８４】（サーバシミュレータの動作）つぎに、サーバシミュレータの動作について説明する。このサーバシミュレータの動作は、上述したサーバシミュレータ起動前処理およびサーバシミュレータ起動処理に相当するものである。図４は、本発明にかかる分散オブジェクト開発システムのサーバシミュレータの動作を示すフローチャートである。特に、ここでは、ＷＷＷブラウザ１８上でサーバシミュレータの制御をおこなう場合について説明する。

【００８５】図４において、まず、アプリケーション開発者は、分散オブジェクト環境１２の下にあるノードの一つ（多くはパーソナルコンピュータである）を使用して、ＷＷＷブラウザ１８を起動させる。そして、シミュレータマシン１０の機能を実現するプログラム（以下、シミュレートプログラムと呼ぶ）をＨＴＭＬ（HyperText Markup Language）として提供するＷＷＷサーバ２０にアクセスするために、そのＷＷＷサーバ２０のアドレスを示すＵＲＬ（Uniform Resource Locator）をＷＷＷブラウザ１８に入力する。これにより、ＷＷＷブラウザ１８上に、シミュレータマシン１０の機能を選択するためのメニュープログラムが起動する（ステップＳ４０１）。

【００８６】図５は、ステップＳ４０１において起動されたメニュープログラムの起動メニュー画面を示す説明図である。図５において、上記したＵＲＬは、アドレス入力フィールド５０５に入力される。図中においては、「http：／／test／bin ／menu.cgi」が入力された状態が示されており、特に、シミュレートプログラムがＣＧＩを介した外部プログラムとしてＷＷＷブラウザ１８と接続されることがわかる。

【００８７】また、図５に示すＷＷＷブラウザ１８では、流通している従来のＷＷＷブラウザのＧＵＩ（Graphical User Interface）と同様に、ＷＷＷページ戻りボタン５０１、ＷＷＷページ進むボタン５０２、アクセス中止ボタン５０３およびＷＷＷページ更新ボタン５０４等が配置されており、マウス等のポインティングデバイスを用いることで各ボタンやその他の選択項目を選択および確定することができ、また、キーボードを用いることで文字入力を可能とする。

【００８８】図５に示すように、シミュレートプログラムによって提供されるメニューは、後述する情報データの作成をおこなうための「Simulation Data Editor」５１０、クライアントシミュレータを起動するための「Client Simulator」５２０およびサーバシミュレータを起動するための「Server Simulator」５３０の選択項目から構成される。

【００８９】なお、これら選択項目は、各項目に応じた処理を示すページへとリンクされており、ＷＷＷブラウザ１８の表示画面において、実際にはこのようなページの更新がおこなわれる毎に、アドレス入力フィールド５０５に表示されるアドレスが変更されるが、以下の説明において参照される表示画面中では、そのアドレスの表

示を省略する。

【００９０】本発明にかかる分散オブジェクト開発システムにおいては、サーバシミュレータを起動させる前に、そのサーバシミュレータを自動的に構築するために、情報ファイルの作成を必要としているため、まず、図５に示した画面上において、「Simulation Data Editor」５１０を選択する。

【００９１】情報ファイルの作成処理は、サーバシミュレータとクライアントシミュレータで共通の入力フォームを与えるため、ここでは、どちらのシミュレータにおける情報ファイルの作成をおこなうかを指示する必要がある。この指示は、図５において、「Simulation Data Editor」５１０を２つの選択項目、たとえば、「Client Simulation Data Editor 」と「Server Simulation Data Editor 」とに分けて表示し、これら２者間の選択をおこなうようにしてもよいし、「Simulation Data Editor」５１０の選択確定後に、新たにサーバシミュレータモードかクライアントシミュレータモードかの選択を促す選択項目を表示してもよく、設計上適宜変更できるものである。

【００９２】ここでは、サーバシミュレータの起動を目的とするため、サーバシミュレータモードの選択をおこなうものとする（ステップＳ４０２）。この選択により、復帰情報作成処理へと処理が移行する（ステップＳ４０３）。ここで、復帰情報とは、上記した情報ファイルをサーバシミュレータモードにおいて作成する場合のその情報ファイルの呼び名である。具体的には、この復帰情報は、サーバからクライアントへ送信される返信情報を示し、この場合では、サーバシミュレータがクライアントアプリケーションに対して返信する情報を意味する。

【００９３】（復帰情報作成処理）ここで、ステップＳ４０３における復帰情報作成処理について説明する。この復帰情報作成処理は、図２のステップＳ２０７におけるサーバシミュレータ起動前処理に相当する処理でもある。図６は、ステップＳ４０３におけるサーバシミュレータの復帰情報作成処理を示すフローチャートである。また、図７は、情報ファイル（ここでは復帰情報を意味する）作成処理における初期入力画面を示す説明図である。

【００９４】図７に示す画面上において、まず、情報ファイル名入力フィールド７１０に、復帰情報ファイル名を入力する（ステップＳ６０１）。この復帰情報ファイル名とは、これから作成する復帰情報の格納先となるファイルの名前であり、たとえば「/home/apts/dat/apt0001_sv.dat」のようなものである。また、この復帰情報ファイル名は、後述するサーバシミュレータの起動時において参照される。

【００９５】つづいて、オブジェクト名入力フィールド７２０に、サーバオブジェクト名を入力する（ステップＳ６０２）。このオブジェクト名とは、クライアントアプリケーションによって呼び出され、かつサーバシミュレータ上でシミュレートされるターゲットオブジェクトの名前であり、たとえば「IDL:apts0001/apts0001:1.0 」のようなものである。

【００９６】そして、「Build 」ボタン７０１を選択することにより、上記したターゲットオブジェクトのＩＤＬ情報の取得がおこなわれ（ステップＳ６０３）、取得した情報が表示されて各データの編集を可能とした状態となる（ステップＳ６０４）。また、すでに復帰情報ファイルが存在している場合は、その復帰情報ファイル名を情報ファイル名入力フィールドに入力するとともに、「Edit」ボタン７０２を選択することで、その復帰情報ファイルの表示および編集が可能となる。また、図７中、「Back」７０３ボタンを選択することにより、図５に示したメニュー画面に戻ることもできる。

【００９７】ここで、本発明にかかる分散オブジェクト開発システムの動作の理解を容易にするために、ターゲットオブジェクトのＩＤＬ定義を図８に示すような構成として、以下の説明をつづける。図８に示すＩＤＬは、後述するクライアントシミュレータについての説明の際も共通に用いるものとする。

【００９８】なお、ステップＳ６０３におけるＩＤＬ情報の取得は、以下の手順にて自動的におこなわれる。まず、ネーミングサービスにアクセスして、上記したオブジェクト名のオブジェクト・リファレンスを取得する。つぎに、取得したオブジェクト・リファレンスを用いて、インプリメンテーションリポジトリにアクセスし、そのターゲットオブジェクトの実装状態を確認する。そして、上記したオブジェクト・リファレンスを用いてインタフェースリポジトリにアクセスし、そこからターゲットオブジェクトのＩＤＬ情報を取得する。

【００９９】図９は、ステップＳ６０４において表示される情報ファイル（ここでは復帰情報を意味する）の編集画面を示す説明図である。図９に示すように、情報ファイル編集画面では、情報ファイル名表示フィールド９１０に、図７の情報ファイル名入力フィールド７１０に入力された情報ファイル名が表示され、また、オブジェクト名表示フィールド９１１に、図７のオブジェクト名入力フィールド７２０に入力されたオブジェクト名が表示される。また、インタフェース名表示フィールド９１２に、実際にネーミングサービスに登録されているターゲットオブジェクトの名前を示すインタフェース名が表示される。

【０１００】そして、図８に示したＩＤＬに従って、図６のステップＳ６０３において取得したＩＤＬ情報として、オペレーション名表示フィールド９２０に「ｏｐ００」が表示され、戻り値属性表示フィールド９２１に「void」が表示される。また、同様に、パラメタ名表示フィールド９３０に「p2」が表示され、パラメタ属性表

示フィールド９３１に「long」が表示され、パラメタ値表示フィールド９３３に「-1」が表示される。

【０１０１】ここで、図示するように、記数方式選択部９３２によって、パラメタ値表示フィールド９３３に表示する記数方式を変更することができ、図９では１０進記数方式を示す「Decimal 」が選択されている。また、戻り値属性表示フィールド９２１に「long」等の戻り値を必要とする属性が表示される場合は、表示フィールド９２２に、記数方式選択部９３２およびパラメタ値表示フィールド９３３と同様な記数方式選択部および戻り値表示フィールドが表示される。

【０１０２】また、図９において、シミュレート繰り返し数入力フィールド９４０に、数値を入力することによって、後述するシミュレートをその入力した数値の示す回数だけ連続して繰り返すことができる。

【０１０３】そして、図９に示した情報ファイル編集画面の表示確認または編集を終えたい場合は、「Save」ボタン９５１を選択することにより、画面上に表示された内容が情報ファイル（この場合、復帰情報）として、情報ファイル名表示フィールド９１０に表示された情報ファイル名で保存される。すなわち、これにより、図６のステップＳ６０５において、復帰情報ファイルが作成される。

【０１０４】なお、復帰情報ファイルには、オブジェクト名、ネーミングサービス登録名、オペレーション名、戻り値名、戻り値属性、戻り値（戻り値属性に合わせて自動生成または編集される）、パラメタ名、パラメタ属性、およびパラメタ値（パラメタ属性に合わせて自動生成または編集される）が保存される。

【０１０５】また、図９において、「Simulator 」ボタン９５２を選択することで、つづいてシミュレータ起動処理（ここでは、サーバシミュレータ）に移行することができ、また、「Back」ボタン９５３を選択することにより、図７に示した初期入力画面に戻ることもできる。

【０１０６】以上に説明したように、復帰情報作成処理によって、クライアントアプリケーションが呼び出すターゲットオブジェクトのＩＤＬ情報をＯＲＢ上から動的に取得することができるので、アプリケーション開発者がこのＩＤＬ情報を従来のようにテキストイメージでのテストデータとしてあらかじめ作成する作業を必要としなくなる。また、取得したＩＤＬ情報を、オペレーション、名前、属性またはパラメタ毎にフィールドを設けてWWWブラウザ１８上に表示しており、アプリケーション開発者による編集を容易にすることができる。

【０１０７】つづいて、図４に示すサーバシミュレータの動作の説明において、ステップＳ４０３の復帰情報作成処理後は、図９に示した「Simulator 」ボタン９５２の選択、または図５に示した「Server Simulator」５３０の選択によって、サーバシミュレータが起動される（ステップＳ４０４）。

【０１０８】図１０は、サーバシミュレータの初期入力画面を示す説明図であり、ステップＳ４０４におけるサーバシミュレータの起動指示によってWWWブラウザ１８上に表示される画面を示している。ここで、サーバシミュレータの起動指示が、図９に示した「Simulator 」ボタン９５２の選択によるものである場合は、復帰情報ファイルの作成が完了しているため、図１０に示すように、その復帰情報ファイル名が、情報ファイル名入力フィールド１０１０に自動的に表示される。

【０１０９】また、同じ理由から、サーバ名表示フィールド１０２０に、図９のオブジェクト名表示フィールド９１１に表示されたオブジェクト名が、サーバ名として表示される。一方、サーバシミュレータの起動指示が、図５に示した「Server Simulator」５３０の選択によるものである場合は、それぞれ復帰情報ファイル名およびサーバ名を入力する必要がある。

【０１１０】さらに、サーバシミュレータの初期入力画面では、検証用ファイル名入力フィールド１０３０に、検証用ファイル名を入力する必要がある（ステップＳ４０５）。ここで、検証用ファイルとは、後述するように、起動情報および実行履歴を一括して保存したファイルであり、サーバシミュレータの実行終了後、この検証用ファイルの内容を調査することで、クライアントアプリレーションが正しく実行されたかどうかを検証することができる。

【０１１１】つづいて、「Execute 」ボタン１０４１を選択することで、サーバシミュレータは、クライアントアプリケーションからの送信待ちの状態に移行する（ステップＳ４０６）。ステップＳ４０６においてクライアントアプリケーションから送信、すなわちターゲットオブジェクトの呼び出しリクエストが発生した場合（図３のステップＳ３０５に相当）は、復帰情報の動的編集の受け付け状態に移行する（ステップＳ４０７）。

【０１１２】復帰情報の動的編集とは、サーバシミュレータの起動後において動的に、上述した復帰情報作成処理をおこなうことであり、復帰情報の内容を、クライアントアプリケーションから送信されたリクエストに応じて動的に変更することができる。

【０１１３】ステップＳ４０７において動的編集の要求がない場合、または動的編集の処理が完了すると、保存された復帰情報ファイルが示す復帰情報を取り込み（ステップＳ４０８）、取り込んだ復帰情報をクライアントアプリケーションに返信する（ステップＳ４０９）。

【０１１４】そして、クライアントアプリケーションが送信したリクエストの内容（オブジェクト名、ネーミングサービス登録名、オペレーション名、パラメタ名、パラメタ値、区切りのキーワード）を起動情報として保存する（ステップＳ４１０）。この際、起動情報において、パラメタの型や値にＩＤＬに従わないような誤りが存在する場合には、その旨を後述する実行履歴に出力す

る。

【０１１５】なお、この起動情報は、クライアントシミュレータの起動情報として再利用することができる。また、サーバシミュレータの実行状態（トレース状態）を実行履歴（実行日時、復帰情報ファイル名、ヘッダ、開始のキーワード、オブジェクト名、オペレーション名、サーバインタフェース動的実装状態、終了状態、終了のキーワード）として保存する（ステップＳ４１１）。

【０１１６】つぎに、アプリケーション開発者からの最新情報表示の要求をチェックする（ステップ４１２）。ここで、最新情報表示の要求とは、不定期にまたは連続的に実行されるクライアントアプリケーションの複数のリクエストに対して、上記した起動情報の表示および実行履歴の表示を直ちにおこなうことを指示することを意味し、図１０における「View」ボタン１０４２の選択によって実現される。

【０１１７】ステップＳ４１２において、最新情報表示の要求が生じなかった場合は、後述するステップＳ４１６の終了要求のチェックをおこなう。また、ステップＳ４１２において、最新情報表示の要求が発生した場合は、図１０における起動情報表示フィールド１０６０に、ステップＳ４１０において保存された起動情報の表示をおこなう（ステップＳ４１３）。図１１は、この起動情報の表示形態を説明する説明図である。図１１に示すように、起動情報は、サーバシミュレータの起動から停止に至るまでに受け取ったクライアントアプリケーションの複数のリクエスト毎に、その内容を起動情報表示フィールド１０６０に表示する。

【０１１８】なお、図中、例として、点線で囲まれた部分、すなわちリクエストＡおよびＢの内容が実際に起動情報表示フィールド１０６０に表示される部分であり、他のリクエストＣの内容の表示は、起動情報表示フィールド１０６０をスクロール表示させることにより可能となる。

【０１１９】クライアントアプリケーションが送信するリクエストには、複数のターゲットオブジェクトの呼び出しを含めることが可能であるため、各ターゲットオブジェクトの呼び出しをそれぞれ独立したリクエストとすることもでき、この起動情報ファイルの単位でかつターゲットオブジェクト毎のリクエストを、起動情報表示フィールド１０６０の下方から上方に向けて表示していく。

【０１２０】図１２は、上記した起動情報の表示例を示す説明図である。図１２に示すように、起動情報表示フィールド１０６０には、オブジェクト名１２０２、ネーミングサービス登録名１２０３、オペレーション名１２０４、パラメタ名およびパラメタ値１２０５、区切りのキーワード１２０７が表示される。また、図示するように、見出し１２０１や復帰情報ファイル名１２０７を表示するようにしてもよい。復帰情報ファイル名１２０７をマウスカーソル等でクリックすることにより、復帰情報作成処理へと移行し、図９に示した編集画面が表示されて、復帰情報ファイルの編集をおこなうようにすることもできる。

【０１２１】つぎに、ステップＳ４１３における起動情報の表示につづいて、図１０における実行履歴表示フィールド１０５０に、ステップＳ４１１において保存された実行履歴の表示をおこなう（ステップＳ４１４）。図１３は、この実行履歴の表示形態を説明する説明図である。図１３に示すように、実行履歴は、図１１に示した起動情報の表示に対応させて、サーバシミュレータの起動から停止に至るまでに受け取ったクライアントアプリケーションの複数のリクエスト毎に、その内容を実行履歴表示フィールド１０５０に表示する。

【０１２２】なお、図中、例として、点線で囲まれた部分、すなわちリクエストＡおよびＢの内容が実際に実行履歴表示フィールド１０５０に表示される部分であり、他のリクエストＣの内容の表示は、実行履歴表示フィールド１０５０をスクロール表示させることにより可能となる。

【０１２３】また、図１１に示したように、実行履歴ファイルの単位でかつターゲットオブジェクト毎のリクエストを、実行履歴表示フィールド１０５０の下方から上方に向けて表示していく。

【０１２４】図１４は、上記した実行履歴の表示例を示す説明図である。図１２に示すように、実行履歴表示フィールド１０５０には、実行日時１４０１、復帰情報ファイル名１４０２、ヘッダ（見出し）１４０３、開始のキーワード１４０４、オブジェクト名１４０５、オペレーション名１４０６、サーバインタフェース動的実装状態１４０７および１４０８、終了状態１４０９、終了のキーワード１４１０が表示される。

【０１２５】そして、ステップＳ４１４における実行履歴の表示につづき、検証用ファイルの作成をおこなう（ステップＳ４１５）。図１５は、検証用ファイルの内部構成を説明する説明図である。図１５に示すように、検証用ファイルは、ステップＳ４１１において作成された実行履歴の内容と、ステップＳ４１０において作成された起動情報の内容（ターゲットオブジェクト毎のリクエスト内容）を単位として、上述したように起動情報ファイルの単位毎にかつクライアントアプリケーションが送信したリクエスト単位毎に編成し、ステップＳ４０５において入力した検証用ファイル名で保存する。

【０１２６】すなわち、検証用ファイルは、サーバシミュレータの起動から停止に至るまでに受け取ったクライアントアプリケーションの複数のリクエスト毎に、図１３に示した実行履歴と図１１に示した起動情報とを順に対にして構成された内容を有して作成される。

【０１２７】そして、サーバシミュレータの実行を終了させる要求をチェックする（ステップＳ４１６）。な

お、この要求は、図１０における「Quit」ボタン１０４３の選択によって実現される。ステップＳ４１６において、終了要求が発生しない場合は、再びステップＳ４０３の復帰情報作成処理に戻り、上述した処理を繰り返す。

【０１２８】また、ステップＳ４１６において、終了要求が生じた場合は、サーバシミュレータの動作を停止する。この場合、クライアントアプリケーション開発者は、開発したクライアントアプリケーションの実行をも終了させる。そして、最終的に、クライアントアプリケーション開発者は、上述した検証用ファイルを参照することにより、開発したクライアントアプリケーションの設計の誤りを検証する。

【０１２９】（サーバアプリケーションの開発手順）つぎに、図２８に示した従来のサーバアプリケーション開発手順に対応させて、本発明にかかる分散オブジェクト開発システムにおけるクライアントシミュレータを用いたサーバアプリケーションの開発手順について説明する。図１６は、本発明にかかる分散オブジェクト開発システムによるサーバアプリケーション開発手順を示すフローチャートである。図１６に示すフローチャートは、図２と同様に、静的起動をおこなうサーバアプリケーションの開発をおこなう手順について示している。

【０１３０】図１６において、まず、サーバアプリケーション開発者は、開発しようとするアプリケーションのプログラム・インタフェース設計をおこなう（ステップＳ１６０１）。つづいてサーバアプリケーションが備えた関数のＩＤＬファイルを作成する（ステップＳ１６０２）。

【０１３１】そして、上記したＩＤＬファイルを用いて実行される所望のサーバアプリケーションのソースを作成する（ステップＳ１６０３）。つぎに、ステップＳ１６０２において作成したＩＤＬファイルを、ＩＤＬコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびＩＤＬオブジェクトを生成する（ステップＳ１６０４）。生成されたＩＤＬオブジェクトは、インタフェースリポジトリに結合される（ステップＳ１６０５）。

【０１３２】つづいて、ステップＳ１６０３において作成されたサーバアプリケーションのソースをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップＳ１６０４において生成されたサーバスケルトンをリンクすることで、実行可能なサーバアプリケーションのファイルを得る（ステップＳ１６０６）。

【０１３３】ステップＳ１６０６において得られたサーバアプリケーションは、インプリメンテーションリポジトリに記録されるとともに、クライアントシミュレータによって呼び出し可能なオブジェクト・インプリメンテーションとして登録される（ステップＳ１６０７）。

【０１３４】そして、上記したクライアントシミュレータを起動する前処理を実行する（ステップＳ１６０８）。このクライアント起動前処理は、本発明にかかる分散オブジェクト開発システムによって実現される処理の一部に相当し、この処理の詳細については後述する。

【０１３５】その後、ステップＳ１６０６において実行可能なファイルとして生成されたサーバアプリケーションを実行させる（ステップＳ１６０９）。ここで実行されるサーバアプリケーションは、分散オブジェクト環境に実際に搭載する形態と同一のコードを有するものであり、特にテスト用のために変更を加える必要はない。すなわち、ＣＯＲＢＡ上で問題なく動作可能なバイナリソースのサーバアプリケーションを実行させる。

【０１３６】図１７は、このサーバアプリケーションの実行処理を説明するためのフローチャートであり、特に静的起動をおこなうためのサーバスケルトンおよびオブジェクトアダプタがおこなう処理を示している。図１７において、まずサーバアプリケーションは、初期化がおこなわれて、オブジェクトアダプタが、クライアントアプリケーション、特にクライアントスタブからオブジェクトの呼び出しリクエストを受け付ける（ステップＳ１７０１）。

【０１３７】受け取ったリクエストにはターゲットオブジェクトのオブジェクト・リファレンスが含まれており、オブジェクトアダプタは、このオブジェクト・リファレンスを用いて、インプリメンテーションリポジトリからターゲットオブジェクトの実装状態を取得する必要がある。

【０１３８】ただし、インプリメンテーションリポジトリもまたＣＯＲＢＡ上のオブジェクトの一つであるため、あらかじめこのインプリメンテーションリポジトリのオブジェクト・リファレンスを取得する必要がある（ステップＳ１７０２）。これにより、インプリメンテーションリポジトリへのアクセスが可能となり、ターゲットオブジェクト（広義ではサーバ）の活性情報の取得が可能となる（ステップＳ１７０３）。

【０１３９】そして、オブジェクトアダプタは、ステップＳ１７０３において取得した活性情報から、サーバが活性済みであるか否かを判定する（ステップＳ１７０４）。ステップＳ１７０４においてサーバが活性されていない場合は、サーバを活性する（ステップＳ１７０５）。サーバの活性後またはステップＳ１７０４においてサーバが活性済みであった場合は、サーバスケルトンが、ターゲットオブジェクトの呼び出しをおこない（ステップＳ１７０６）、ターゲットオブジェクトから返信データを受け取るとともに、この返信データを発信元であるクライアントアプリケーションに送出する（ステップＳ１７０７）。

【０１４０】この返信データは、ＯＲＢコアを介して再び、クライアントスタブによってリクエスト発信元のクライアントアプリケーションに届けられるが、本発明に

かかる分散オブジェクト開発システムでは、この返信データ受け取り処理およびリクエスト送出処理を、上記したクライアントシミュレータがおこなう。よって、クライアントシミュレータは、換言すれば、クライアントスタブの機能をシミュレートすることにもなる。

【０１４１】ステップＳ１７０７の返信処理後、オブジェクトアダプタは、サーバの活性化をおこなった場合、そのサーバの非活性化等の後処理を実行する（ステップＳ１７０８）。

【０１４２】図１６において、サーバアプリケーションを実行させた後は、クライアントシミュレータを起動する処理を実行する（ステップＳ１６１０）。このクライアントシミュレータ起動処理もまた、本発明にかかる分散オブジェクト開発システムによって実現される処理の一部に相当し、この処理の詳細についても後述する。

【０１４３】そして、クライアントシミュレータが送信したリクエストと、そのリクエストに対してサーバアプリケーションが返信した返信データと、によって、サーバアプリケーションの動作結果の検証がおこなわれる（ステップＳ１６１１）。

【０１４４】（クライアントシミュレータの動作）つぎに、クライアントシミュレータの動作について説明する。このクライアントシミュレータの動作は、上述したクライアントシミュレータ起動前処理およびクライアントシミュレータ起動処理に相当するものである。図１８は、本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの動作を示すフローチャートである。特に、ここでは、WWWブラウザ１８上でクライアントシミュレータの制御をおこなう場合について説明する。

【０１４５】図１８において、まず、アプリケーション開発者は、分散オブジェクト環境１２の下にあるノードの一つを使用して、WWWブラウザ１８を起動させる。そして、図５に示したメニュープログラムを起動させる（ステップＳ１８０１）。そして、クライアントシミュレータ用の情報ファイル、すなわち起動情報を作成するため、図５に示した画面上において、「Simulation Data Editor」５１０を選択する。なお、情報ファイルの作成処理は、上述したサーバシミュレータの説明においておこなったものと同様であるので、ここではその説明を省略する。

【０１４６】よって、上述したような選択手段により、クライアントシミュレータモードの選択をおこなう（ステップＳ１８０２）。この選択により、起動情報作成処理へと処理が移行する（ステップＳ１８０３）。ここで、起動情報とは、上記した情報ファイルをクライアントシミュレータモードにおいて作成する場合のその情報ファイルの呼び名である。具体的には、この起動情報は、クライアントからサーバへ送信されるリクエストを示す。

【０１４７】（起動情報作成処理）つぎに、ステップＳ１８０３における起動情報作成処理について説明する。なお、この起動情報作成処理は、図１６のステップＳ１６０８におけるクライアントシミュレータ起動前処理に相当する処理でもある。図１９は、ステップＳ１８０３におけるクライアントシミュレータの起動情報作成処理を示すフローチャートである。なお、情報ファイル（ここでは起動情報を意味する）作成処理における初期入力画面については図７に示した図と同様であるため、これを利用する。

【０１４８】図７において、まず、情報ファイル名入力フィールド７１０に、起動情報ファイル名を入力する（ステップＳ１９０１）。この起動情報ファイル名とは、これから作成する起動情報の格納先となるファイルの名前であり、たとえば「/home/apts/dat/apt0001__cl.dat」のようなものである。また、この起動情報ファイル名は、後述するクライアントシミュレータの起動時において参照される。

【０１４９】つづいて、オブジェクト名入力フィールド７２０に、サーバオブジェクト名を入力する（ステップＳ１９０２）。このオブジェクト名とは、サーバアプリケーションが受け付け、かつクライアントシミュレータ上で送信されるリクエストに含まれるターゲットオブジェクトの名前であり、たとえば「IDL:apts0001/apts0001:1.0 」のようなものである。

【０１５０】そして、「Build 」ボタン７０１を選択することにより、上記したターゲットオブジェクトのＩＤＬ情報の取得がおこなわれ（ステップＳ１９０３）、取得した情報が表示されて各データの編集を可能とした状態となる（ステップＳ１９０４）。また、すでに起動情報ファイルが存在している場合は、その起動情報ファイル名を情報ファイル名入力フィールドに入力するとともに、「Edit」ボタン７０２を選択することで、その起動情報ファイルの表示および編集が可能となる。

【０１５１】ステップＳ１９０３におけるＩＤＬ情報の取得は、図６のステップＳ６０３と同様であり、ステップＳ１９０４において表示される情報ファイル（ここでは起動情報を意味する）の編集画面もまた、図６のステップＳ６０４と同様であるので、ここではこれらの説明を省略する。

【０１５２】よって、この起動情報作成処理により、オブジェクト名、ネーミングサービス登録名、オペレーション名、戻り値名、戻り値属性、パラメタ名、パラメタ属性およびパラメタ値が保存された起動情報ファイルが作成される（ステップＳ１９０５）。

【０１５３】なお、図９に示した「Simulator 」ボタン９５２を選択することで、つづいてシミュレータ起動処理（ここでは、クライアントシミュレータ）に移行することができる。

【０１５４】以上に説明したように、起動情報作成処理

によって、サーバアプリケーションが受け付けるターゲットオブジェクトのＩＤＬ情報をＯＲＢ上から動的に取得することができるので、アプリケーション開発者がこのＩＤＬ情報を従来のようにテキストイメージでのテキストデータとしてあらかじめ作成する作業を必要としなくなる。また、取得したＩＤＬ情報を、オペレーション、名前、属性またはパラメタ毎にフィールドを設けてＷＷＷブラウザ１８上に表示しており、アプリケーション開発者による編集を容易にすることができる。

【０１５５】つづいて、図１８に示すサーバシミュレータの動作の説明において、ステップＳ１８０３の復帰情報作成処理後は、図９に示した「Simulator 」ボタン９５２の選択、または図５に示した「Client Simulator」５２０の選択によって、クライアントシミュレータが起動される（ステップＳ１８０４）。

【０１５６】図２０は、クライアントシミュレータの初期入力画面を示す説明図であり、ステップＳ１８０４におけるクライアントシミュレータの起動指示によってＷＷＷブラウザ１８上に表示される画面を示している。ここで、クライアントシミュレータの起動指示が、図９に示した「Simulator 」ボタン９５２の選択によるものである場合は、起動情報ファイルの作成が完了しているため、図２０に示すように、その起動情報ファイル名が、情報ファイル名入力フィールド２０１０に自動的に表示される。

【０１５７】一方、サーバシミュレータの起動指示が、図５に示した「Client Simulator」５２０の選択によるものである場合は、起動情報ファイル名を入力する必要がある。

【０１５８】さらに、クライアントシミュレータの初期入力画面では、検証用ファイル名入力フィールド２０２０に、検証用ファイル名を入力する必要がある（ステップＳ１８０５）。ここで、検証用ファイルとは、上述したように、起動情報および実行履歴を一括して保存したファイルであり、クライアントシミュレータの実行終了後、この検証用ファイルの内容を調査することで、サーバアプリレーションが正しく実行されたかどうかを検証することができる。

【０１５９】つづいて、「Try 」ボタン２０４１を選択することで、クライアントシミュレータは、保存された起動情報ファイルが示す起動情報を取り込み（ステップＳ１８０６）、取り込んだ起動情報をリクエストとしてサーバアプリケーションに送信する（ステップＳ１８０７）。

【０１６０】そして、サーバアプリケーションからの返信待ちの状態に移行する（ステップＳ１８０８）。ステップＳ１８０８においてサーバアプリケーションから返信があった場合（図１７のステップＳ１７０７に相当）は、その返信の内容（オブジェクト名、ネーミングサービス登録名、オペレーション名、パラメタ名、パラメタ値、区切りのキーワード）を復帰情報として保存する（ステップＳ１８０９）。この際、復帰情報において、パラメタの型や値にＩＤＬに従わないような誤りが存在する場合には、その旨を後述する実行履歴に出力する。

【０１６１】なお、この復帰情報ファイルは、サーバシミュレータの復帰情報として再利用することができる。また、クライアントシミュレータの実行状態（トレース状態）を実行履歴（実行日時、復帰情報ファイル名、ヘッダ、開始のキーワード、オブジェクト名、オペレーション名、サーバインタフェース動的実装状態、リクエスト送信、返信受信、終了状態、終了のキーワード）として保存する（ステップＳ１８１０）。

【０１６２】そして、図２０における起動情報表示フィールド２０６０に、ステップＳ１８０９において保存された復帰情報の表示をおこなう（ステップＳ１８１１）。図２１は、この復帰情報の表示形態を説明する説明図である。図２１に示すように、復帰情報は、クライアントシミュレータの起動から停止に至るまでに受け取ったサーバアプリケーションの複数の返信毎に、その内容を復帰情報表示フィールド２０６０に表示する。

【０１６３】なお、図中、例として、点線で囲まれた部分の内容が実際に復帰情報表示フィールド２０６０に表示される部分であり、他の内容の表示は、復帰情報表示フィールド２０６０をスクロール表示させることにより可能となる。この返信内容の表示は、サーバアプリケーションからの返信を受け取る毎に、その返信内容を復帰情報として復帰情報表示フィールド２０６０の下方から上方に向けて表示していく。

【０１６４】図２２は、上記した復帰情報の表示例を示す説明図である。図２２に示すように、復帰情報表示フィールド２０６０には、オブジェクト名２２０１、ネーミングサービス登録名２２０２、オペレーション名２２０３、パラメタ名およびパラメタ値２２０４、区切りのキーワード２２０５が表示される。

【０１６５】つぎに、ステップＳ１８１１における復帰情報の表示につづいて、図２０における実行履歴表示フィールド２０５０に、ステップＳ１８１０において保存された実行履歴の表示をおこなう（ステップＳ１８１２）。図２３は、この実行履歴の表示形態を説明する説明図である。図２３に示すように、実行履歴は、図２１に示した復帰情報の表示に対応させて、クライアントシミュレータの起動から停止に至るまでに受け取ったサーバアプリケーションの返信毎に、その内容を実行履歴表示フィールド２０５０に表示する。

【０１６６】なお、図中、例として、点線で囲まれた部分の内容が実際に実行履歴表示フィールド２０５０に表示される部分であり、他の返信内容の表示は、実行履歴表示フィールド２０５０をスクロール表示させることにより可能となる。また、図２１に示したように、この実行履歴は、クライアントシミュレータのリクエスト送信

毎に、実行履歴表示フィールド２０５０の下方から上方に向けて表示していく。

【０１６７】図２４は、上記した実行履歴の表示例を示す説明図である。図２４に示すように、実行履歴表示フィールド２０５０には、実行日時２３０１、起動情報ファイル名２３０２、ヘッダ（見出し）２３０３、開始のキーワード２３０４、オペレーション名２３０５、サーバインタフェース動的実装状態２３０６および２３０７、リクエスト送信２３０８、返信受信２３０９、終了状態２３１０、終了のキーワード２３１１が表示される。

【０１６８】そして、ステップＳ１８１２における実行履歴の表示につづき、検証用ファイルの作成をおこなう（ステップＳ１８１３）。図２５は、検証用ファイルの内部構成を説明する説明図である。図２５に示すように、検証用ファイルは、ステップＳ１８１０において作成された実行履歴の内容と、ステップＳ１８０９において作成された復帰情報の内容（サーバアプリケーションからの返信内容）を単位として編成し、ステップＳ１８０５において入力した検証用ファイル名で保存する。

【０１６９】すなわち、検証用ファイルは、クライアントシミュレータの起動から停止に至るまでに受け取ったサーバアプリケーションの返信毎に、図２３に示した実行履歴と図２１に示した起動情報とを順に対にして構成された内容を有して作成される。

【０１７０】そして、クライアントシミュレータの実行を終了させる要求をチェックする（ステップＳ１８１４）。なお、この要求は、図２０における「Quit」ボタン２０４２の選択によって実現される。ステップＳ１８１４において、終了要求が発生しない場合は、再びステップＳ１８０３の起動情報作成処理に戻り、上述した処理を繰り返す。

【０１７１】また、ステップＳ１８１４において、終了要求が生じた場合は、クライアントシミュレータの動作を停止する。この場合、サーバアプリケーション開発者は、開発したサーバアプリケーションの実行をも終了させる。そして、最終的に、サーバアプリケーション開発者は、上述した検証用ファイルを参照することにより、開発したサーバアプリケーションの設計の誤りを検証する。

【０１７２】以上に説明したように、実施の形態にかかる分散オブジェクト開発システムおよび分散オブジェクト開発方法によれば、ＩＤＬ情報を取得することで、クライアントアプリケーションの実行検証をおこなう際にテスト用サーバアプリケーションを必要とする場合やサーバアプリケーションの実行検証をおこなう際にテスト用クライアントアプリケーションを必要とする場合に準備されるテストデータ（情報ファイル）を、取得したＩＤＬ情報を利用して自動的に作成するので、このようなテストデータの作成を手作業でおこなうことなく、アプリケーション開発者の負担を軽減させることができる。

【０１７３】また、このように作成したテストデータを用いてオブジェクトの呼び出しおよび／またはオブジェクトの実行をおこなうことにより、サーバアプリケーションまたはクライアントアプリケーションをシミュレートすることができるので、テスト用アプリケーションの開発を必要とせず、アプリケーション開発に要する時間を短縮させることが可能となる。

【０１７４】WWWサーバ２０上においてサーバシミュレータまたはクライアントシミュレータを自動的に構築することで、クライアントおよびサーバアプリケーションの開発環境を分離するとともに、WWWブラウザ１８上を介して、サーバシミュレータまたはクライアントシミュレータの動作を制御できるので、所望のホストシステム上にアプリケーション開発環境を構築することができる。

【０１７５】なお、以上に説明した実施の形態において、図４および図１８に示した処理を実行させるシミュレートプログラムを、ＩＣカードメモリ、フロッピーディスク、光磁気ディスク、ＣＤ－ＲＯＭ等の記録媒体に格納し、この記録媒体に記録されたプログラムを、分散オブジェクト環境１２上のいずれかのノード（シミュレータマシン１０およびWWWサーバ２０を含む）にインストールすることで、上述した分散オブジェクト開発方法を使用し、これにより分散オブジェクト開発システムを構築することもできる。なお、このインストール作業は、通信回線を使用して一方のノードが他方のノードからダウンロードすることによっておこなってもよい。

【０１７６】

【発明の効果】以上、説明したとおり、本発明にかかる分散オブジェクト開発システム（請求項１）、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体（請求項９）によれば、分散オブジェクト環境において、オブジェクトの型情報を定義するとともにクライアントアプリケーションとサーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、オブジェクトの呼び出しおよび／またはオブジェクトの実行をシミュレートするので、アプリケーションの実行検証をおこなう際に対となるテスト用のアプリケーションの実行に必要なテストデータを、従来のように手作業で作成することなく、自動的に生成させて、アプリケーション開発者の負担を軽減させることができ、このテストデータをオブジェクトの呼び出しおよび／またはオブジェクトの実行をおこなう際の通信電文として用いることでサーバシミュレータまたはクライアントシミュレータを自動的に構築することができ、アプリケーション開発に要する時間を短縮させることが可能となるという効果を奏する。

【０１７７】また、本発明にかかる分散オブジェクト開

発システム（請求項２）、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体（請求項１０）によれば、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成し、生成された応答を前記クライアントアプリケーションに対して送信し、また、前記インタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出し要求を生成し、生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するので、サーバアプリケーションおよびクライアントアプリケーションをシミュレートする機能を自動的に構築することができ、アプリケーション開発者の負担の軽減およびアプリケーション開発に要する時間の短縮を図ることができるという効果を奏する。

【０１７８】また、本発明にかかる分散オブジェクト開発システム（請求項３）によれば、複数のノードのうちの少なくとも一つのノードが前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段を備えており、当該ノードを除く他のノードのうちの所望のノードが、前記ネットワークを介して、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を制御するので、テストの対象となるアプリケーションと、シミュレートされるアプリケーションとをネットワーク上に分離して配置することができるという効果を奏する。

【０１７９】また、本発明にかかる分散オブジェクト開発システム（請求項４）によれば、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を、前記ノードの一つであるＷＷＷサーバ上に備え、前記ノードの他の一つがＷＷＷブラウザを介して前記シミュレート機能を制御するので、アプリケーション開発者が、ＷＷＷブラウザ上のＧＵＩ表示による容易な操作を享受することができ、アプリケーション開発の効率を高めることができるとともに、ＷＷＷブラウザを介して、所望のホストシステム上にアプリケーション開発環境を構築することができるという効果を奏する。

【０１８０】また、本発明にかかる分散オブジェクト開発システム（請求項５）によれば、前記サーバアプリケーションシミュレート手段が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容と、前記生成された応答の内容と、から構成される検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したクライアントアプリケーションの設計の誤りを検証することができるという効果を奏する。

【０１８１】また、本発明にかかる分散オブジェクト開発システム（請求項６）によれば、前記サーバアプリケーションシミュレート手段が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された応答の内容に付加して、前記検証用ファイルを作成するので、検証用ファイルを参照することで、開発したクライアントアプリケーションの設計の誤りを容易に発見することができるという効果を奏する。

【０１８２】また、本発明にかかる分散オブジェクト開発システム（請求項７）によれば、前記クライアントアプリケーションシミュレート手段が、前記生成された要求の内容と、該要求に応じて前記サーバアプリケーションから送信された応答の内容と、から構成される検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したサーバアプリケーションの設計の誤りを検証することができるという効果を奏する。

【０１８３】また、本発明にかかる分散オブジェクト開発システム（請求項８）によれば、前記クライアントアプリケーションシミュレート手段が、前記サーバアプリケーションから返信された応答の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された要求の内容に付加して、前記検証用ファイルを作成するので、開発したサーバアプリケーションの設計の誤りを容易に発見することができるという効果を奏する。

【図面の簡単な説明】

【図１】本発明にかかる分散オブジェクト開発システムの概略構成を示すブロック図である。

【図２】本発明にかかる分散オブジェクト開発システムによるクライアントアプリケーション開発手順を示すフローチャートである。

【図３】クライアントアプリケーションの実行処理を説明するためのフローチャートである。

【図４】本発明にかかる分散オブジェクト開発システムのサーバシミュレータの動作を示すフローチャートである。

【図５】本発明にかかる分散オブジェクト開発システムの起動メニュー画面を示す説明図である。

【図６】本発明にかかる分散オブジェクト開発システムのサーバシミュレータの復帰情報作成処理を示すフローチャートである。

【図７】本発明にかかる分散オブジェクト開発システムの情報ファイル作成処理における初期入力画面を示す説明図である。

【図８】本発明にかかる分散オブジェクト開発システム

の説明に用いるＩＤＬを示す図である。
【図９】本発明にかかる分散オブジェクト開発システムの情報ファイルの編集画面を示す説明図である。
【図１０】本発明にかかる分散オブジェクト開発システムのサーバシミュレータの初期入力画面を示す説明図である。
【図１１】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける起動情報の表示形態を説明する説明図である。
【図１２】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける起動情報の表示例を示す説明図である。
【図１３】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける実行履歴の表示形態を説明する説明図である。
【図１４】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける実行履歴の表示例を示す説明図である。
【図１５】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける検証用ファイルの内部構成を説明する説明図である。
【図１６】本発明にかかる分散オブジェクト開発システムによるサーバアプリケーション開発手順を示すフローチャートである。
【図１７】サーバアプリケーションの実行処理を説明するためのフローチャートである。
【図１８】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの動作を示すフローチャートである。
【図１９】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの起動情報作成処理を示すフローチャートである。
【図２０】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの初期入力画面を示す説＊明図である。
【図２１】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける復帰情報の表示形態を説明する説明図である。
【図２２】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける復帰情報の表示例を示す説明図である。
【図２３】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける実行履歴の表示形態を説明する説明図である。
【図２４】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける実行履歴の表示例を示す説明図である。
【図２５】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける検証用ファイルの内部構成を説明する説明図である。
【図２６】ＣＯＲＢＡにおけるＯＲＢのインタフェース構造を示すブロック図である。
【図２７】従来におけるクライアントアプリケーション開発手順を示すフローチャートである。
【図２８】従来におけるサーバアプリケーション開発手順を示すフローチャートである。
【符号の説明】
１０　シミュレータマシン
１０ａ　サーバアプリケーションシミュレーション部
１０ｂ　クライアントアプリケーションシミュレーション部
１０ｃ　プログラムインタフェース実装部
１２　分散オブジェクト環境
１４　ネットワーク
１６　ＩＤＬ定義格納部
１８　WWWブラウザ
２０　WWWサーバ
２２　コマンド部

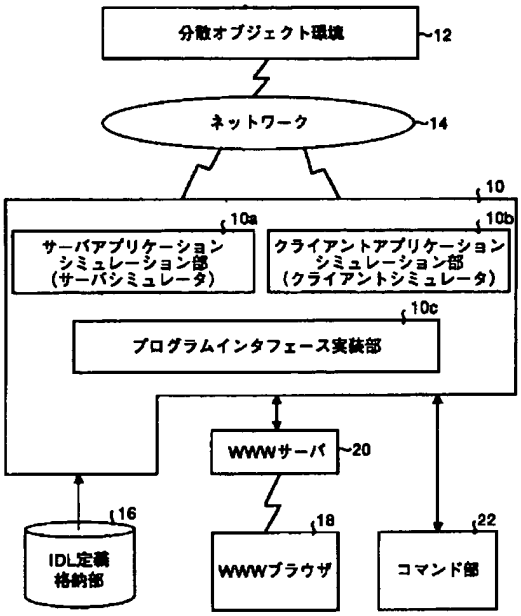
【図８】

本発明にかかる分散オブジェクト開発システムの説明に用いるIDLを示す図

```
module      apts0001 {
            interface apts0001 {
                        void op00 ( in long p1 , out long p2 ) ;
            } ;
} ;
```

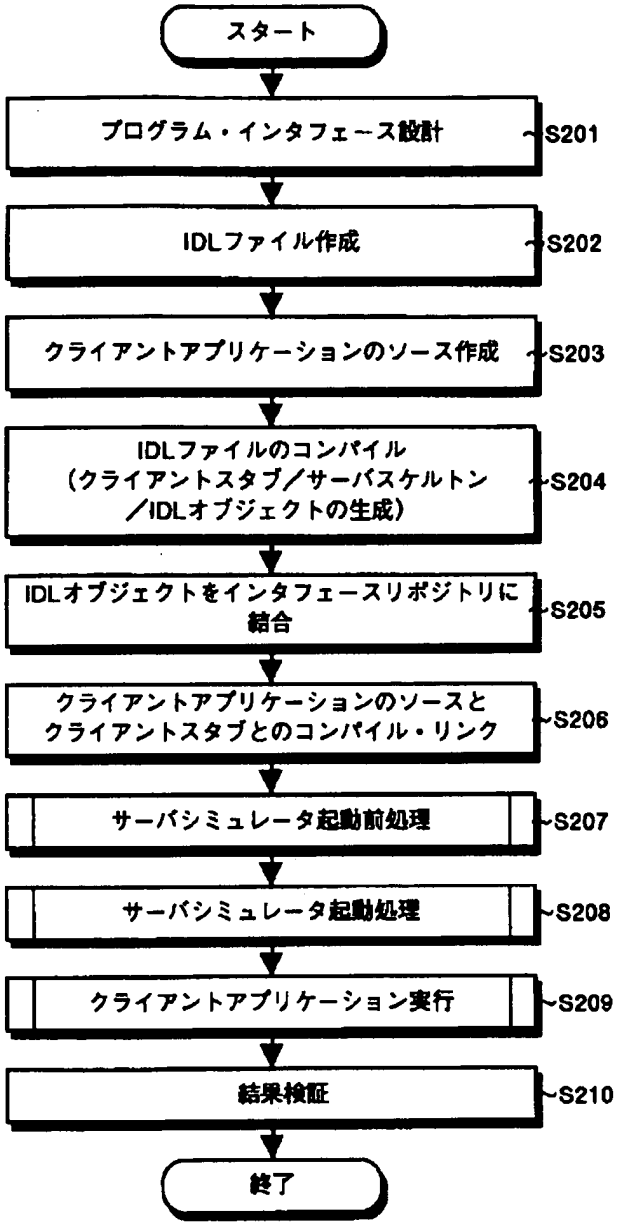
【図１】

本発明にかかる分散オブジェクト開発システムの概略構成図



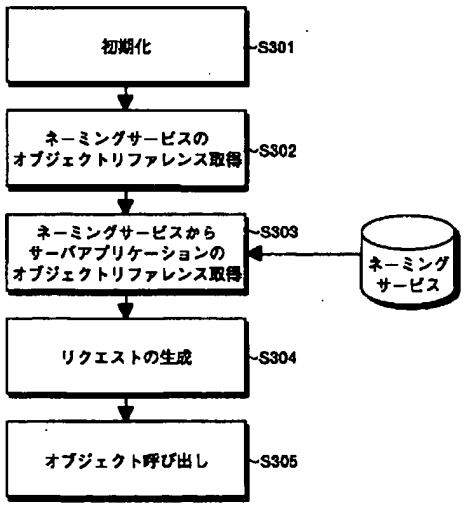
【図２】

本発明にかかる分散オブジェクト開発システムによる
クライアントアプリケーション開発手順を示すフローチャート



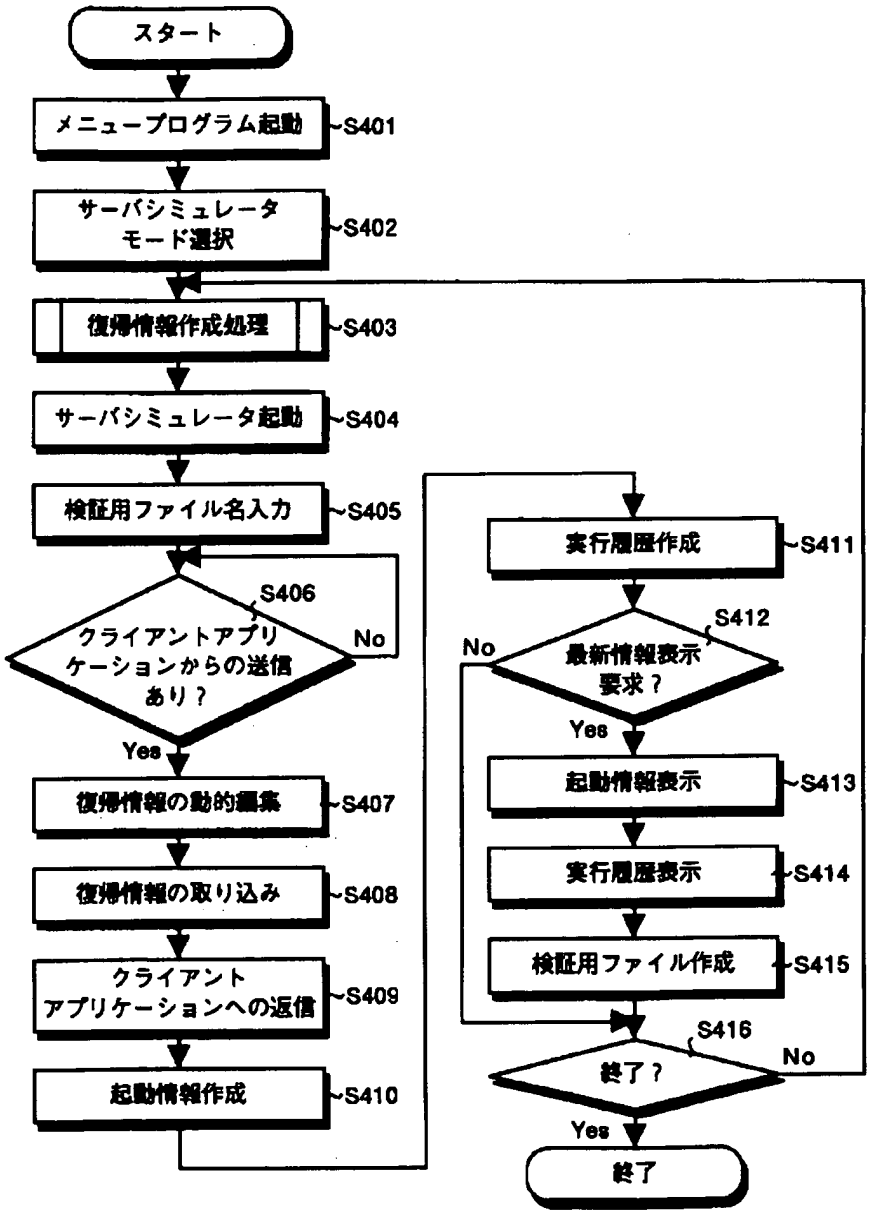
【図３】

クライアントアプリケーションの実行処理を
説明するためのフローチャート



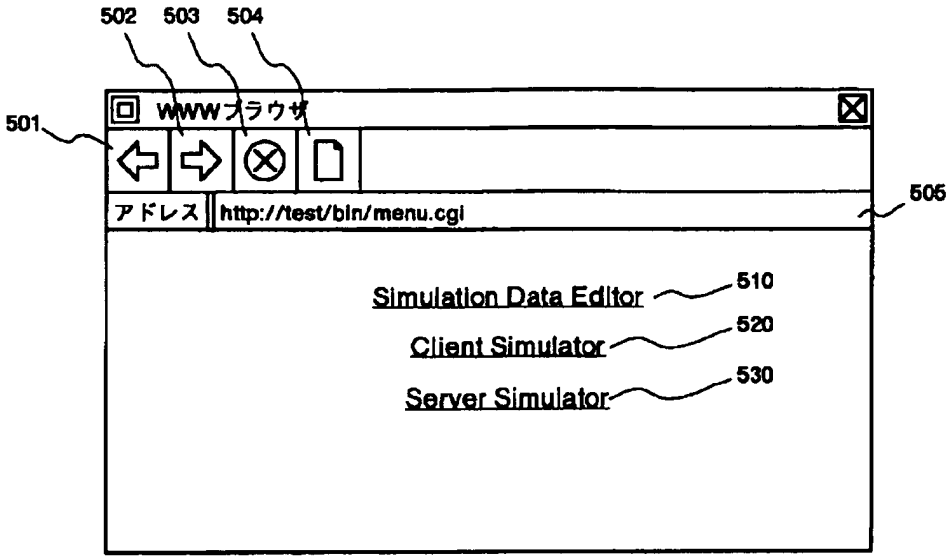
【図４】

本発明にかかる分散オブジェクト開発システムの
サーバシミュレータの動作を示すフローチャート



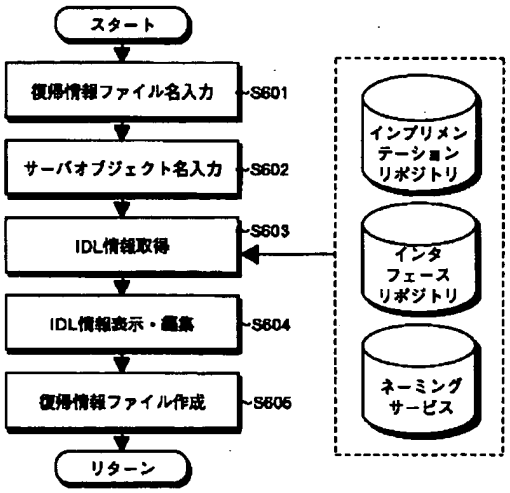
【図５】

起動メニュー画面を示す説明図



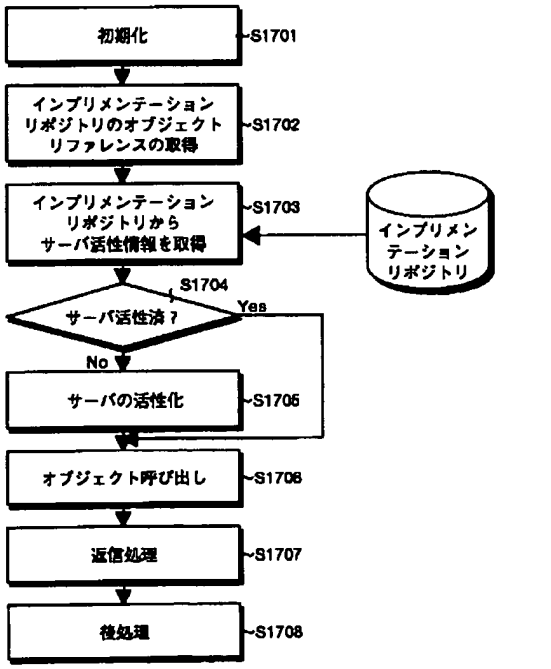
【図６】

サーバシミュレータの復帰情報作成処理を示すフローチャート



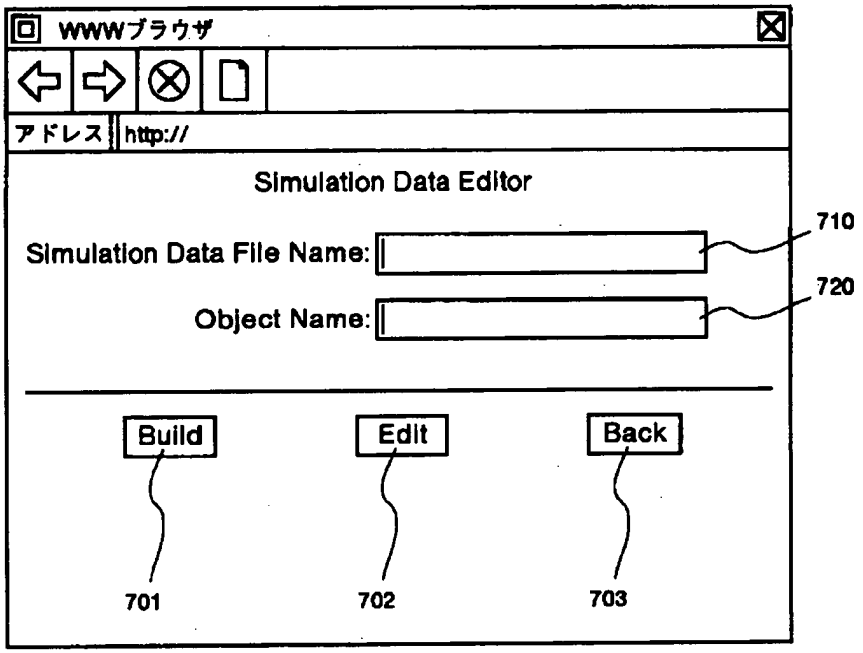
【図１７】

サーバアプリケーションの実行処理を説明するためのフローチャート



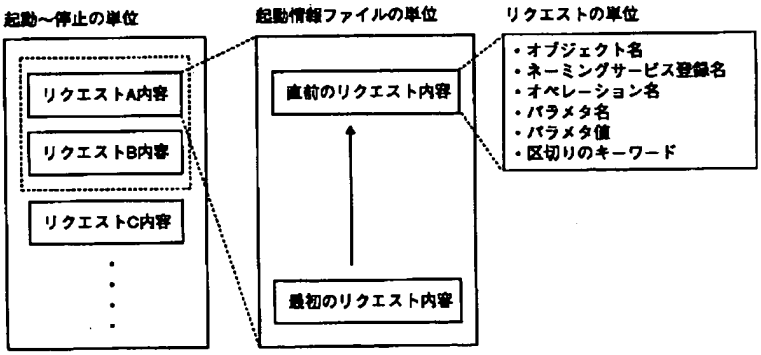
【図７】

情報ファイル作成処理における初期入力画面を示す説明図



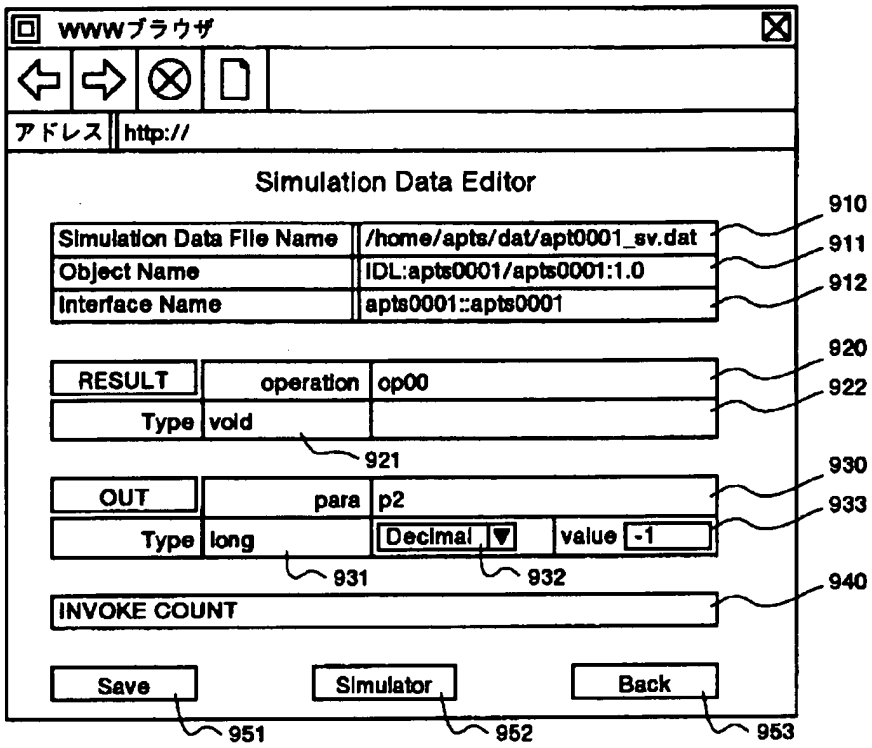
【図１１】

サーバシミュレータにおける起動情報の表示形態を説明する説明図



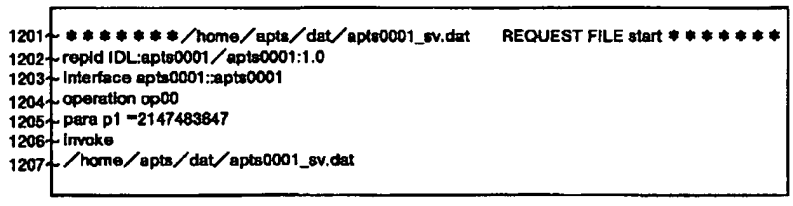
【図９】

情報ファイルの編集画面を示す説明図



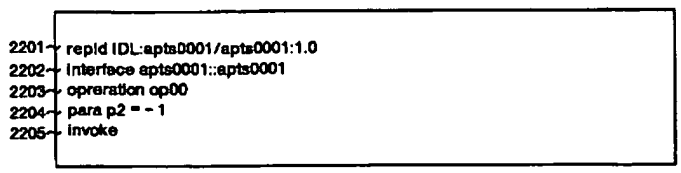
【図１２】

サーバシミュレータにおける起動情報の表示例を示す説明図



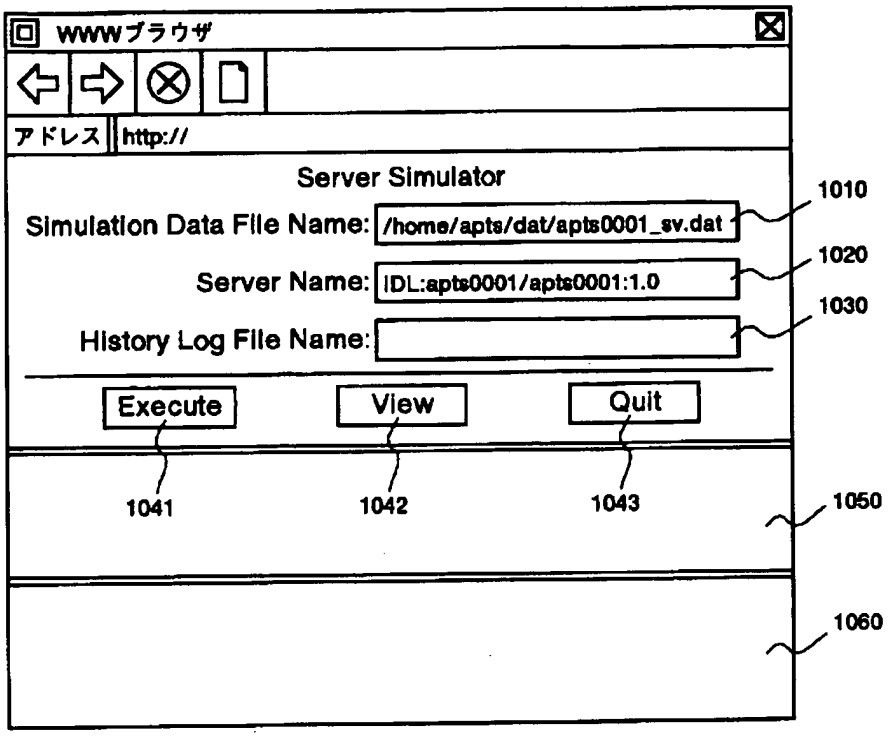
【図２２】

クライアントシミュレータにおける復帰情報の表示例を示す説明図



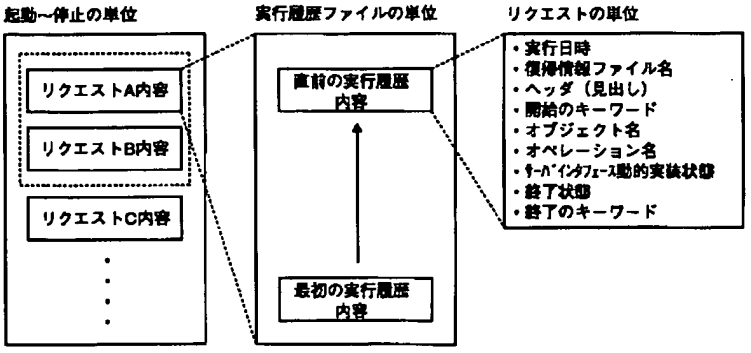
【図１０】

サーバシミュレータの初期入力画面を示す説明図



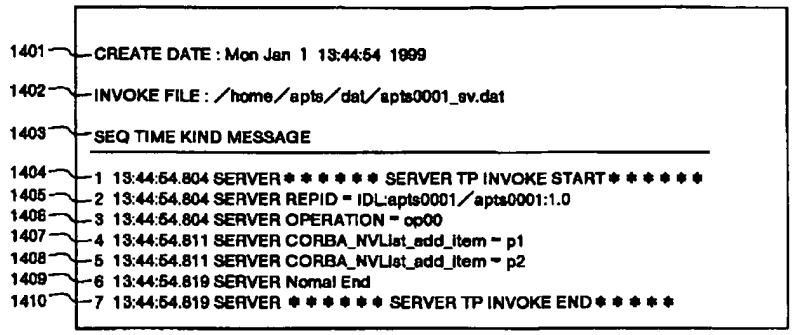
【図１３】

サーバシミュレータにおける実行履歴の表示形態を説明する説明図



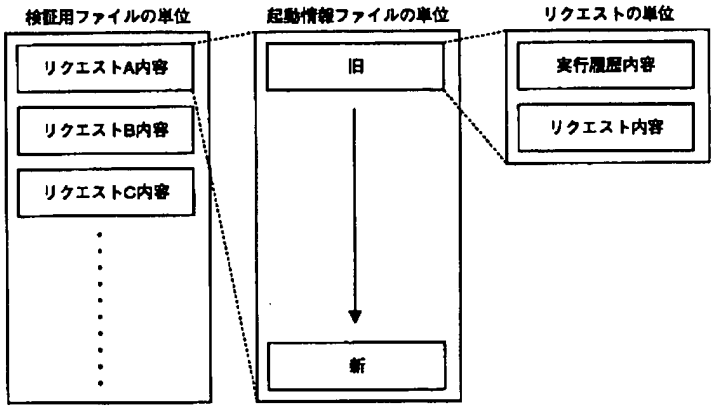
【図１４】

サーバシミュレータにおける実行履歴の表示例を示す説明図



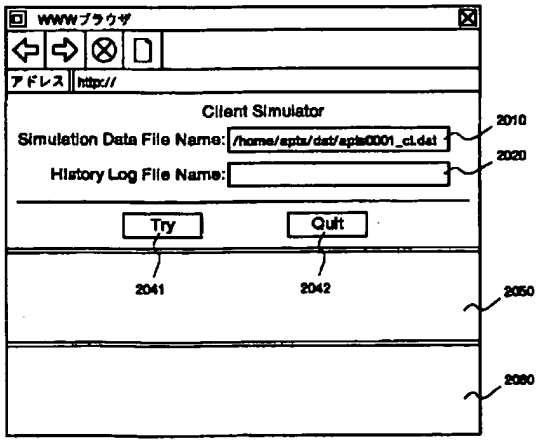
【図１５】

サーバシミュレータにおける検証用ファイルの内部構成を説明する説明図



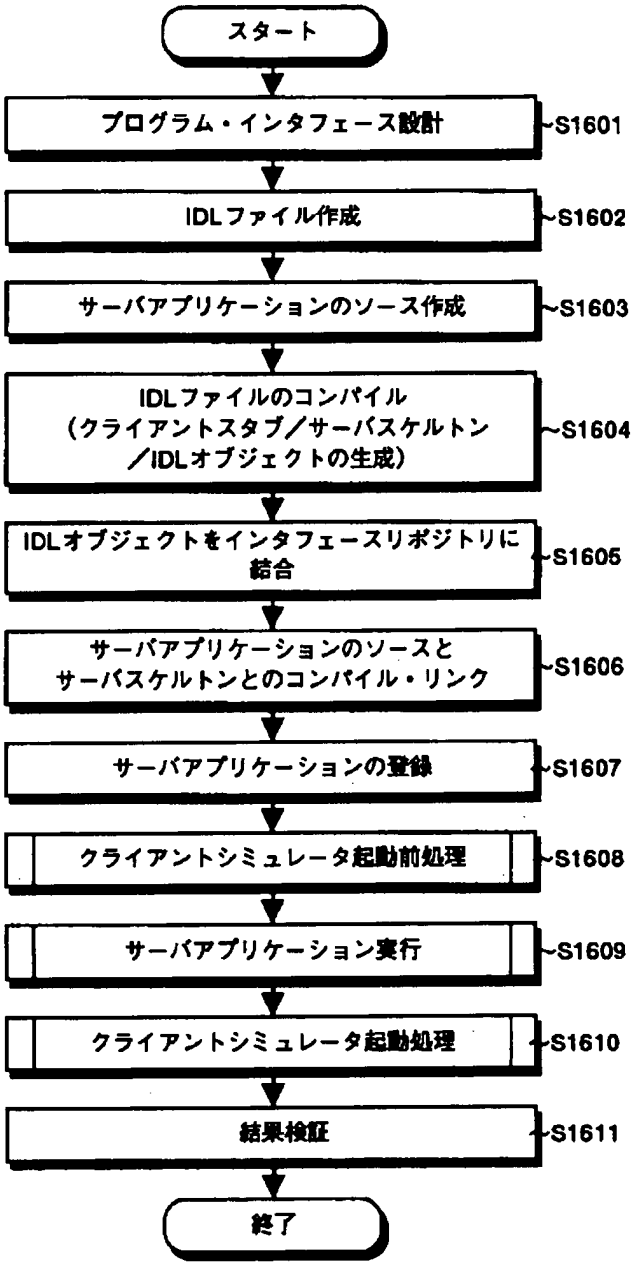
【図２０】

クライアントシミュレータの初期入力画面を示す説明図



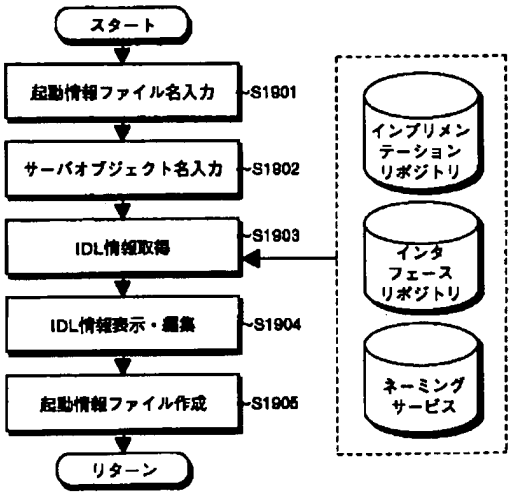
【図１６】

本発明にかかる分散オブジェクト開発システムによるサーバアプリケーション開発手順を示すフローチャート



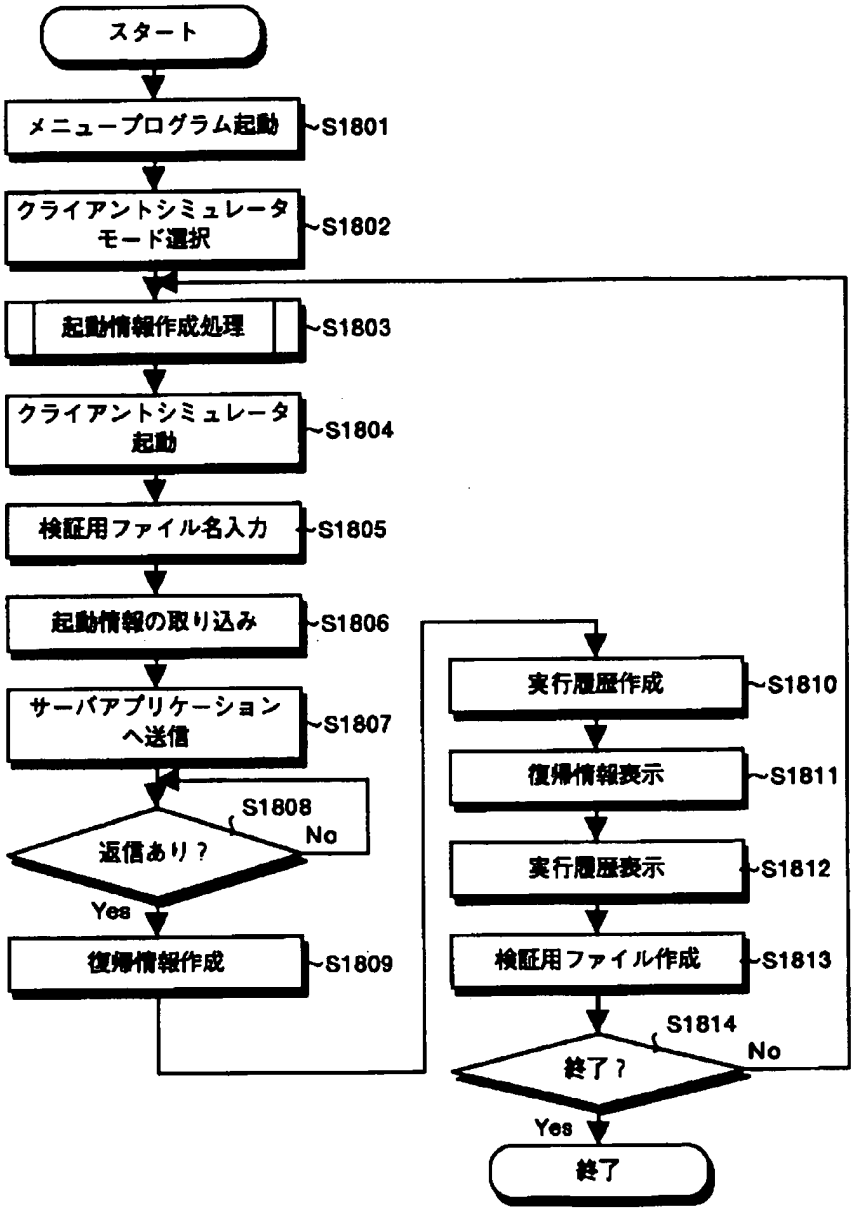
【図１９】

クライアントシミュレータの起動情報作成処理を示すフローチャート



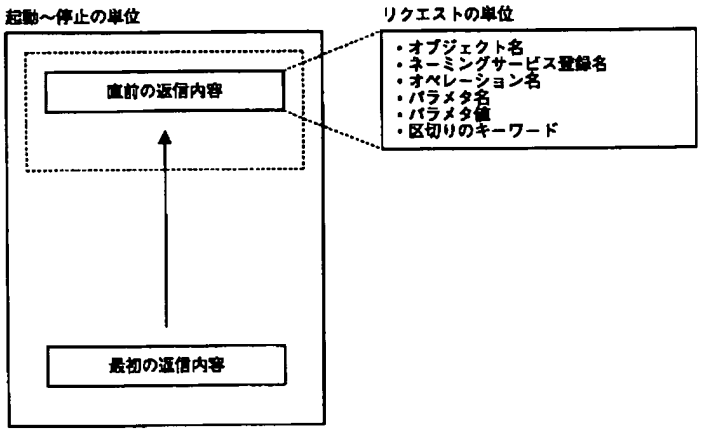
【図１８】

本発明にかかる分散オブジェクト開発システムの
クライアントシミュレータの動作を示すフローチャート



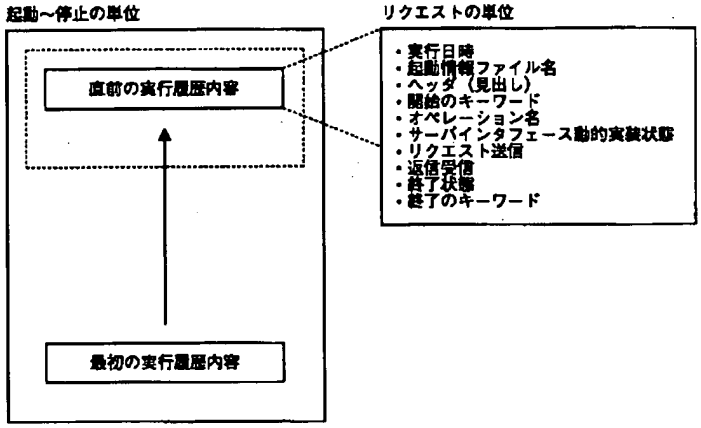
【図２１】

クライアントシミュレータにおける復帰情報の表示形態を説明する説明図



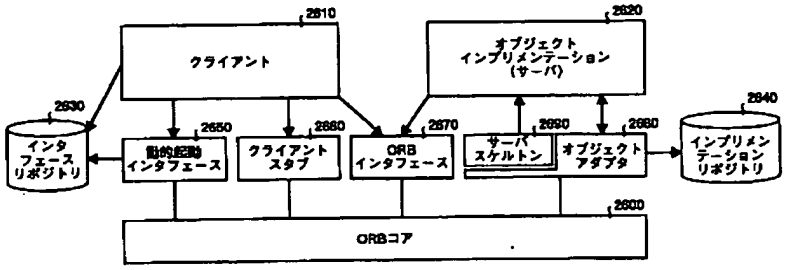
【図２３】

クライアントシミュレータにおける実行履歴の表示形態を説明する説明図



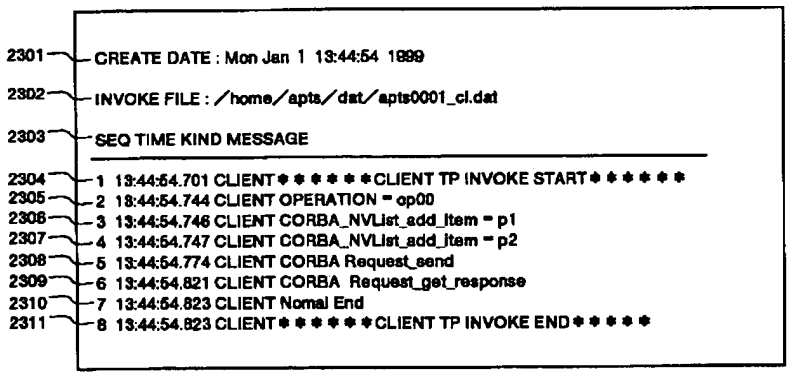
【図２６】

CORBAにおけるORBのインタフェース構造を示すブロック図



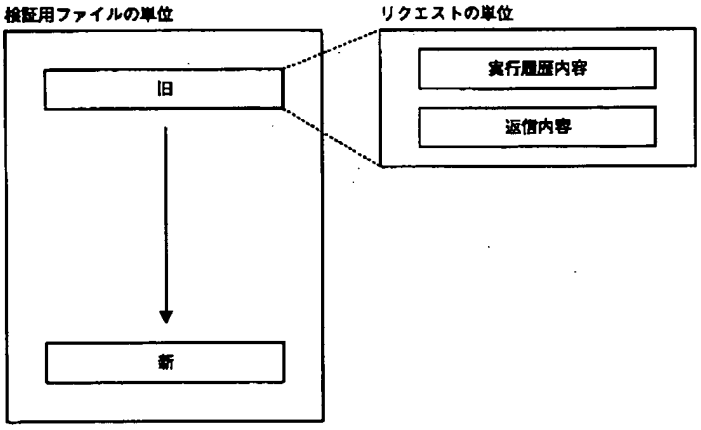
【図２４】

クライアントシミュレータにおける実行履歴の表示例を示す説明図



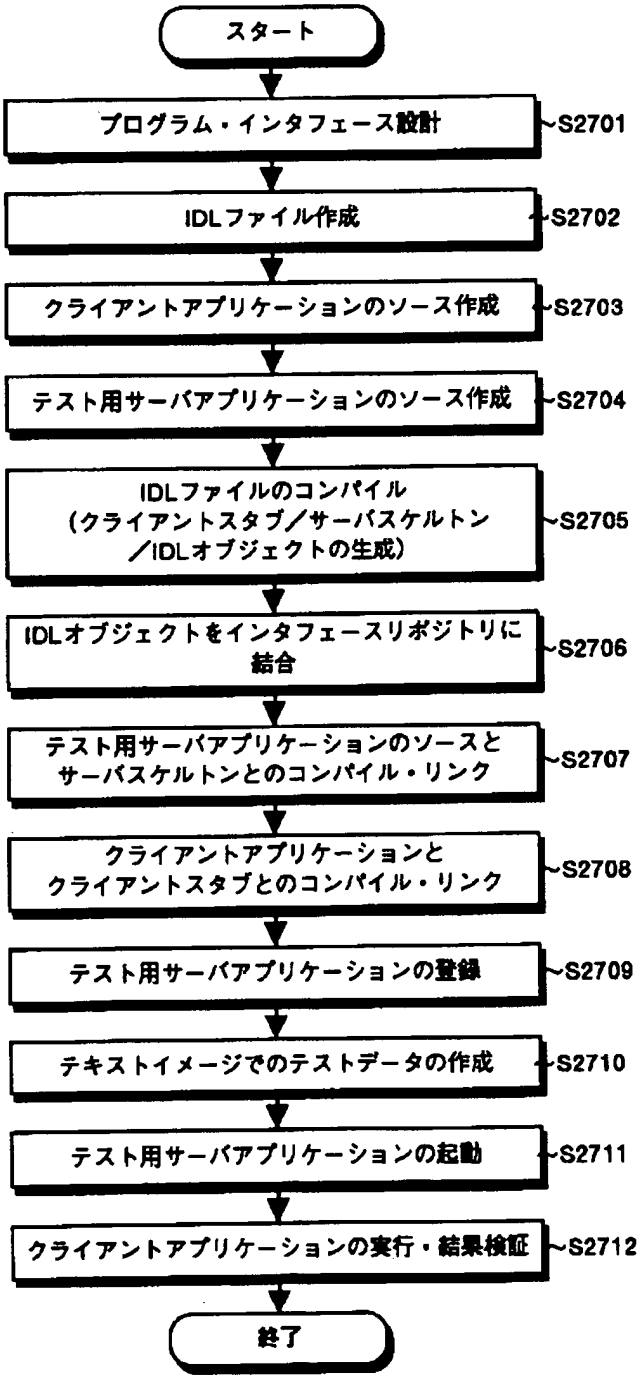
【図２５】

クライアントシミュレータにおける検証用ファイルの内部構成を説明する説明図



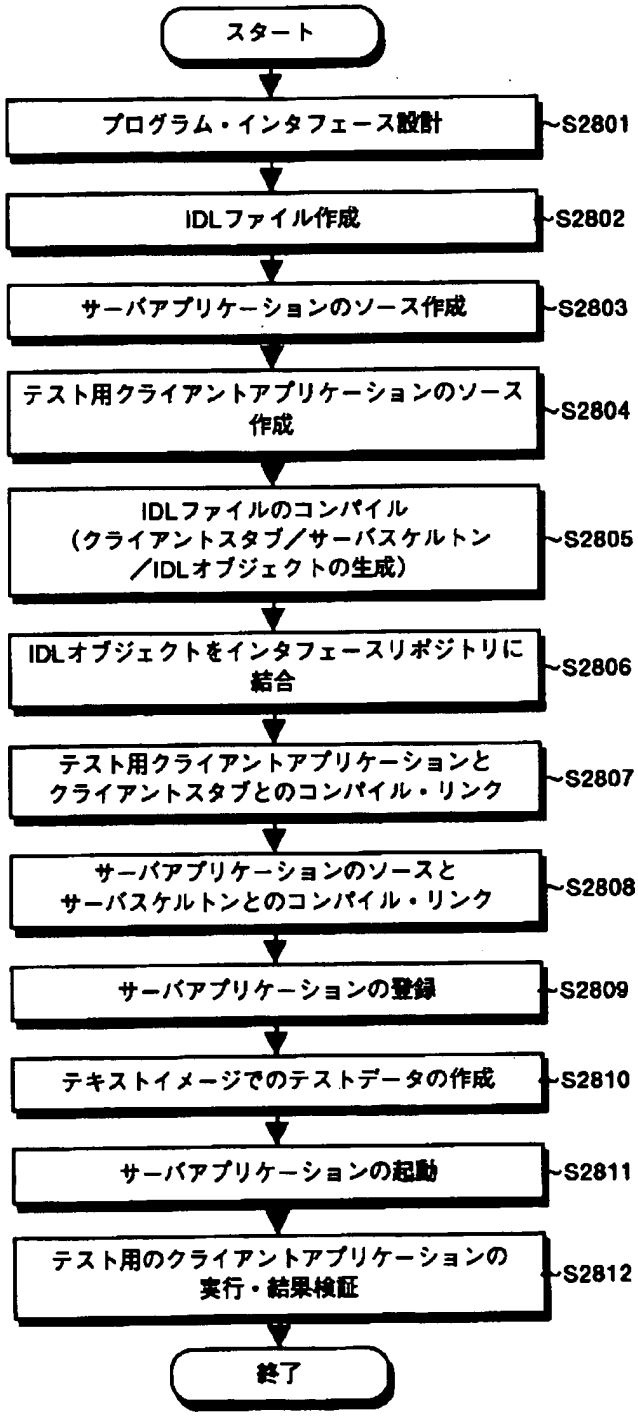
【図２７】

従来のクライアントアプリケーション開発手順を示すフローチャート



【図２８】

従来のサーバアプリケーション開発手順を示すフローチャート



フロントページの続き

Ｆターム(参考) 5B042 GA08 GA12 HH07 HH11
5B045 GG01 KK05
5B076 DD03 EC06